A CONSENSYS DILIGENCE AUDIT REPORT

Balancer Finance

Date	May 2020
Lead Auditor	Alexander Wade
Co-auditors	Shayan Eskandari

1 Executive Summary

In April 2020, Balancer asked us to conduct a security assessment of Balancer Finance - Balancer core: an automated portfolio manager, liquidity provider, and price sensor.

We performed this assessment from May 4 to May 15, 2020. The assessment primarily focused on the high-level logic of balancer-core: `BPool`. The engagement was conducted by Alexander Wade and Shayan Eskandari, the total effort spent was 4 person-weeks.

1.1 Scope

Our review focused on the commit hash `5d70da92b1bebaa515254d00a9e064ecac9bd18e`. The list of files in scope can be found in the Appendix.

Balancer's `BPool` implementation makes use of a set of complicated formulas for interacting with the protocol. The definitions and derivations of these formulas are located in the whitepaper (see below). The EVM implementation of these formulas requires algebraic transformations, exponentiation approximation, and other considerations in order to compute these formulas with reasonable margin of error and gas costs.

The general correctness of these formulas and their implementation was out of scope for this assessment, as the priority for this review was the high-level logic of `BPool` and its parent contracts.

1.2 Documentation

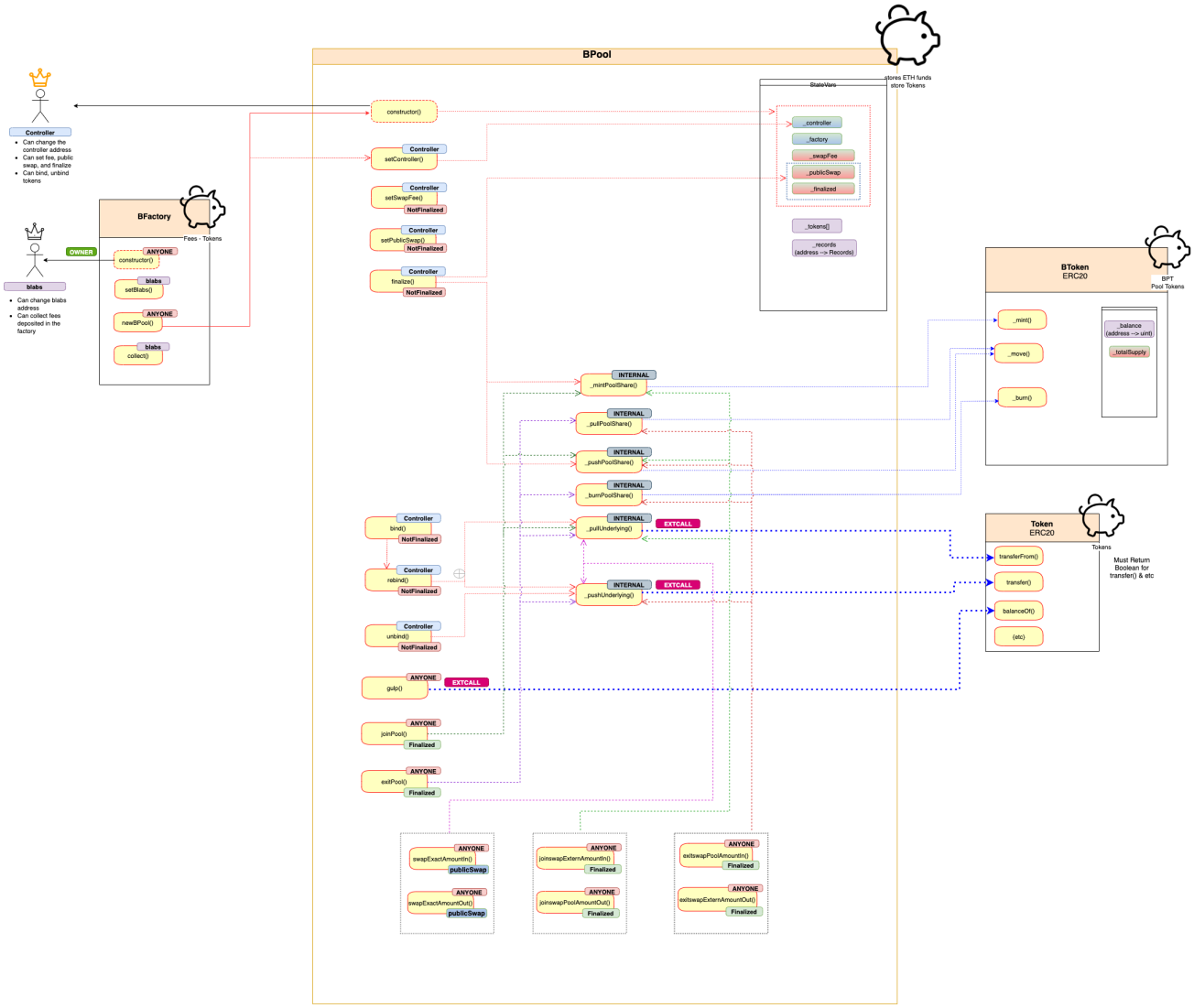
Alongside an initial code walkthrough provided by the client, the following documentation was available during our assessment:

- Whitepaper
- Docs
  - In particular, the sections on Math and Exponentiation approximation are particularly relevant for any traders interested in using Balancer's protocol.
- Inline comments

2 System Overview

Balancer is "a generalized Uniswap" that users can hold tokens in a pool with ratios other than 50-50. The ratios are calculated by the normalized weight of each token in the pool.

Below you can see the visualization of the Balancer system.



3 Recommendations

During the course of our review, we made the following recommendations:

3.1 Restrict access to `setController` so that it may only be called before finalization

Description

`setController` is used to change the privileged `_controller` address, which is able to perform many administrative actions before calling `finalize`. After finalization, the `_controller` serves no purpose.

Locking the function will ensure it is not used, and will reduce confusion for users of the `BPool`.

Recommendation

Add `require(!finalized)` to `BPool.setController`

3.2 Ensure bound and rebound token values are exactly correct

Description

For both `BPool.bind` and `BPool.rebind`, the `balance` parameter is used to determine how many tokens the pool will absorb from `msg.sender` (or release to `msg.sender`):

code/contracts/BPool.sol:L286-L297

```
// Adjust the balance record and actual token balance
uint oldBalance = _records[token].balance;
_records[token].balance = balance;
if (balance > oldBalance) {
    _pullUnderlying(token, msg.sender, bsub(balance, oldBalance));
} else if (balance < oldBalance) {
    // In this case liquidity is being withdrawn, so charge EXIT_FEE
    uint tokenBalanceWithdrawn = bsub(oldBalance, balance);
    uint tokenExitFee = bmul(tokenBalanceWithdrawn, EXIT_FEE);
    _pushUnderlying(token, msg.sender, bsub(tokenBalanceWithdrawn, tokenExit
    _pushUnderlying(token, _factory, tokenExitFee);
}
```

Because token balance changes can happen outside of the context of this function, an extra check at the bottom would ensure that the `rebind` operation was performed successfully and with complete understanding of the state of the pool:

`require(_records[token].balance == token.balanceOf(address(this)));`

Alternatively, consider performing an operation similar to that implemented in `gulp`:

code/contracts/BPool.sol:L333-L341

```
// Absorb any tokens that have been sent to this contract into the pool
function gulp(address token)
    external
    _logs_
    _lock_
{
    require(_records[token].bound, "ERR_NOT_BOUND");
    _records[token].balance = IERC20(token).balanceOf(address(this));
}
```

3.3 Include sanity-check for `extcodesize` on bound tokens

Description

Generally, users of a `BPool` should recognize and trust all of the pool's bound tokens before interacting with it. To help with this somewhat (and ensure addresses are not bound accidentally), an `extcodesize` check could be added to `BPool.bind`.

Recommendation

Ensure `extcodesize` of tokens is nonzero in `BPool.bind`

3.4 Consider implementing a minimum `_totalWeight` for `unbind` and `rebind`

Description

`BPool.rebind` and `BPool.unbind` do not explicitly check that a decrease in `_totalWeight` results in a usable value. Swaps will not function correctly if `_totalWeight` moves outside of certain bounds; the `MAX_TOTAL_WEIGHT` restriction in `rebind` provides some assurance on the cap of `_totalWeight`:

code/contracts/BPool.sol:L276-L280

```
// Adjust the denorm and totalWeight
uint oldWeight = _records[token].denorm;
if (denorm > oldWeight) {
    _totalWeight = badd(_totalWeight, bsub(denorm, oldWeight));
    require(_totalWeight <= MAX_TOTAL_WEIGHT, "ERR_MAX_TOTAL_WEIGHT");
```

Implementing a minimum value will provide assurance on the lower bound of `_totalWeight`.

Recommendation

Add a `require` to `rebind` and `unbind` that `MIN_WEIGHT * _tokens.length <= _totalWeight`

Alternatively, automatically set `_publicSwap` to `false` if `_totalWeight` drops below `MIN_WEIGHT`.

3.5 Disallow self-bound pools

Description

`BPool`'s token can be interacted with in much the same way as the rest of the pool's bound tokens, even if it is not bound. `joinPool`, `exitPool`, `joinswap*`, and `exitswap*` each allow users to purchase and sell a pool's own token in exchange for varying quantities of the pool's bound tokens.

However, `BPool`'s token can also be bound to its own pool explicitly. In this case, many internal accounting functions do not properly track operations (`transfer`, `mint`, `burn`, etc) performed on pool tokens.

Recommendation

Disallow binding a pool's token to itself. Add a check in `bind`:

```
require(token != address(this));
```

3.6 Use of modifiers for repeated checks

Description

It is recommended to use modifiers for common checks within different functions. This will result in less code duplication in the given smart contract and adds significant readability into the code base.

Examples

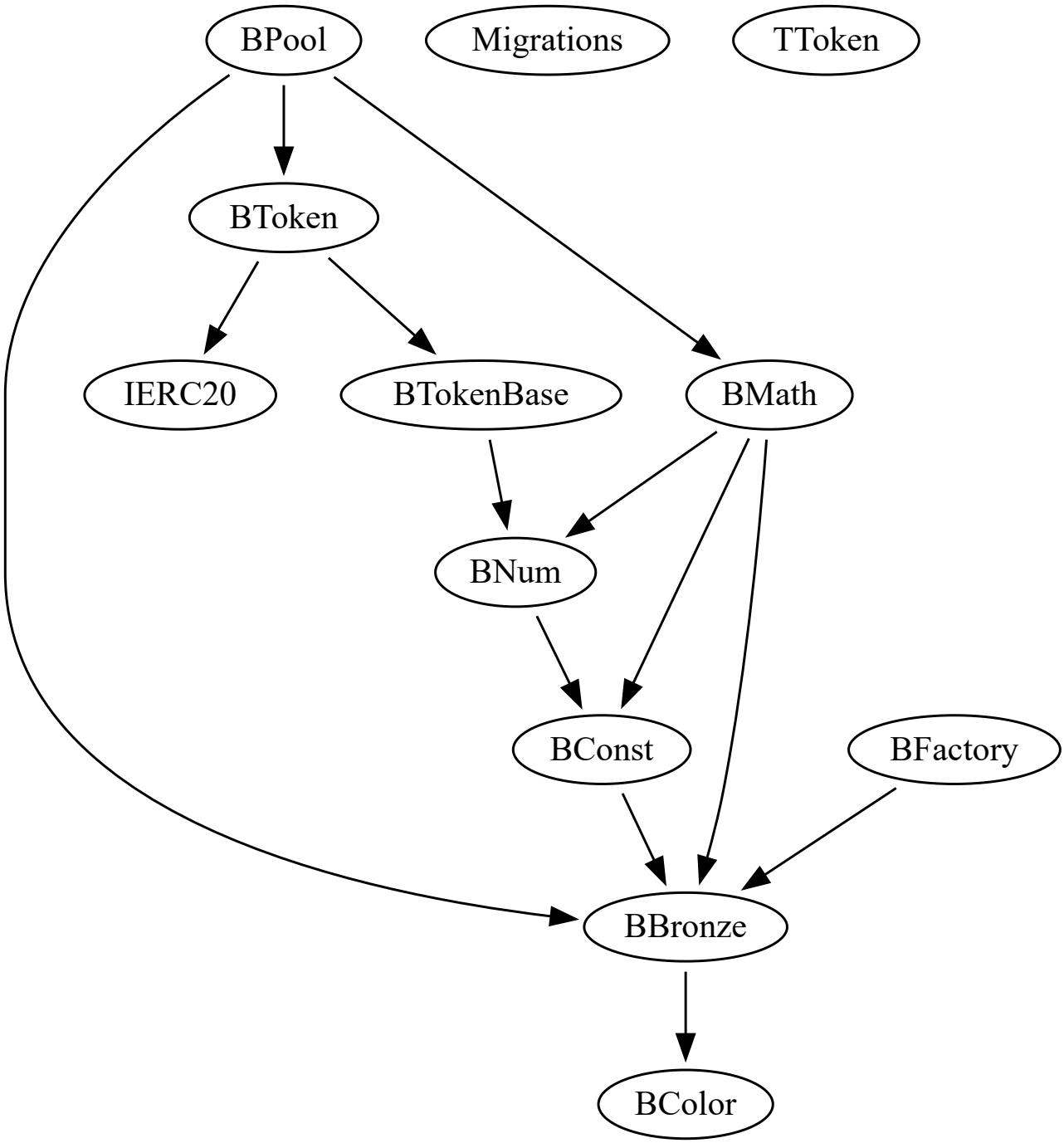
The main suggestion is for, but not limited to, the following checks in `BPool.sol` contract:

- `require(msg.sender == _controller, "ERR_NOT_CONTROLLER");` has been repeated 7 times in `BPool` contract, which can be replaced with `onlyController()` modifier with the same require
- `require(!_finalized, "ERR_IS_FINALIZED");` has been repeated 6 times in the contract, similarly this can be replaced with `notFinalized()` modifier with the same require
- `require(_finalized, "ERR_NOT_FINALIZED");` has been repeated 7 times in the contract, it can be replaced with `finalized()` modifier with the same require

3.7 Remove unused code

Description

`BColor.sol` which includes `BColor` and `BBronze` contracts, solely exist to indicate the version of the factory and the pool. BBronze is inherited in many contracts and makes overall contract structure unnecessary complicated.



Recommendation

The color (version) can be represented by the something like following line in `BConst.sol`:

```
bytes32 public constant BColor = bytes32("BRONZE");
```

3.8 PBT unique naming

Description

Currently each pool mints its own token named `Balancer Pool Token` with the symbol `BPT`. If tracked on etherscan, all pools show the same token name, but different address, which might be confusing to the users.

Examples

Balancer Poo... (BPT)
6,805.87612798 BPT

Balancer Poo... (BPT)
1,986.7169164 BPT

Balancer Poo... (BPT)
1,987.62590459 BPT

Recommendation

Let Pool controller name their Pool share token.

3.9 Inconsistent require checks in AmountIn & AmountOut

Description

The main difference between `*AmountIn` and `*AmountOut` are that one checks the lower bound price using `minAmountOut` and the other the maximum price using `maxPoolAmountIn`, reflectively for “buy” and “sell” tokens.

However, the checks in some of these functions are inconsistent.

Example

code/contracts/BPool.sol:L595-L605

```
tokenAmountIn = calcSingleInGivenPoolOut(
                    inRecord.balance,
                    inRecord.denorm,
                    _totalSupply,
                    _totalWeight,
                    poolAmountOut,
                    _swapFee
                );

require(tokenAmountIn != 0, "ERR_MATH_APPROX");
require(tokenAmountIn <= maxAmountIn, "ERR_LIMIT_IN");
```

The equivalent non-zero check from the above code snippet is missing in the `joinswapExternAmountIn` function below:

code/contracts/BPool.sol:L562-L572

```
poolAmountOut = calcPoolOutGivenSingleIn(
                    inRecord.balance,
                    inRecord.denorm,
                    _totalSupply,
                    _totalWeight,
                    tokenAmountIn,
                    _swapFee
                );

require(poolAmountOut >= minPoolAmountOut, "ERR_LIMIT_OUT");
```

The check happens implicitly by the following line, but none of the checked values had a non-zero check beforehand.

```
require(poolAmountOut >= minPoolAmountOut, "ERR_LIMIT_OUT");
```

Recommendation

Verify all the checks in similar functions.

Also based on the code similarity in the `*AmountIn` and `*AmountOut` functions, there might be a better way to implement these pair functions and merge them together. The solution is yet to be discussed and can be implemented on future versions of Balancer.

3.10 Perform more rigorous input validation across swap functions

Description

Several functions could use additional input validation checks. Generally, many functions tend to allow trades with nonsensical input and output values, which may exposes edge-case behavior.

The following examples provide several locations where additional input validation should be performed:

Examples

1. `joinPool` and `exitPool` should both check that `maxAmountsIn` and `minAmountsOut` have equivalent length to `BPool._tokens`
2. `swapExactAmountIn` and `swapExactAmountOut` should check that `tokenIn != tokenOut`
3. `swapExactAmountIn` and `swapExactAmountOut` should check that both `spotPriceBefore` and `spotPriceAfter` are nonzero.
4. `swapExactAmountIn` should check that `tokenAmountOut != 0`
5. `swapExactAmountOut` should check that `tokenAmountIn != 0`
6. `joinswapExternAmountIn` should check that `tokenAmountIn != 0` and that `poolAmountOut != 0`
7. `joinswapPoolAmountOut` should check that `poolAmountOut != 0`
8. `exitswapPoolAmountIn` should check that `poolAmountIn != 0` and that `tokenAmountOut != 0`
9. `exitswapExternAmountOut` should check that `tokenAmountOut != 0`

Recommendation

Add the aforementioned sanity checks to all trade functions.

Additionally, reject trades where “zero tokens” are either the input or the output.

4 Security Specification

This section describes, **from a security perspective**, the expected behavior of the system under audit. It is not a substitute for documentation. The purpose of this section is to identify specific security properties that were validated by the audit team.

4.1 Actors

The relevant actors are listed below with their respective abilities:

- **BLabs**: *BFactory owner* The address deploying BFactory
  - Can change the *BLabs* address
  - Can collect factory fees from pools
- **Pool Controller**: Each pool has an address associated with it as *Controller*, which is the address calling `newBPool()` in the BFactory contract
  - Can change the controller address
  - Can set *SwapFee*, which is enforced to be between *MIN_FEE* and *MAX_FEE* (Defined in `BConst` as 0.0001% and 10% respectively)
  - Can switch *publicSwap*, given that the pool is *not finalized* yet
  - Can *Finalize* the pool, which will make the pool public and joinable for others
  - Can *bind*, *rebind*, and *unbind* tokens to the pool (up to 8 tokens for each pool), and set the weights of each token. This is only possible when the pool is *not finalized* yet
- **Anyone**: Any other ethereum address
  - Can update the balance of the tokens in the pool by calling `gulp()`
  - Can *Join* and *Exit* any *finalized* pool and deposit tokens based on their max prices
  - Can Swap Pool token, and individual tokens

4.2 Trust Model

In any smart contract system, it's important to identify what trust is expected/required between various actors. For this audit, in addition to Actors section, we established the following trust model:

- It is important for anyone willing to join a pool to make sure all the tokens bound to that pool are recognized and verified. Many functionalities in the pool, such as *Join Pool*, *Exit Pool*, and *Swap* functions, do external calls to the tokens contracts and it is assumed that the bound tokens are safe to interact with.
  - Any upgradable tokens must be verified before each call to the pool.
- Pool Exit fee is currently set to 0 in `BConst.sol`, however the code exist to send the fees to the factory on rebinding tokens or exiting pool.

- On joining the pool, a maximum token amount `maxAmountsIn` is passed to protect user from high price fluctuation that may be caused by front-running or other users. These values should be correctly calculated and visible in the user interface.
- The mathematic formulas implemented in `BMath.sol` and `BNum.sol` follow the formulas in the Balancer whitepaper. However their implementations are restricted by Solidity limits. Same as issue 5.1, more rounding issues might exist and requires further unit tests for edge cases.
- As noted in the documentation, Balancer Pools only supports ERC-20 implementations that return Boolean for transfer(), transferFrom(), and other functionalities.

5 Issues

Each issue has an assigned severity:

- Minor issues are subjective in nature. They are typically suggestions around best practices or readability. Code maintainers should use their own judgment as to whether to address such issues.
- Medium issues are objective in nature but are not security vulnerabilities. These should be addressed unless there is a clear reason not to.
- Major issues are security vulnerabilities that may not be directly exploitable or may require certain conditions in order to be exploited. All major issues should be addressed.
- Critical issues are directly exploitable security vulnerabilities that need to be fixed.

5.1 Similar token-to-token swap methods can yield very different results Medium

Description

`BPool`'s interface exposes several methods to perform token swaps. Because the formula used to calculate trade values varies depending on the method, we compared token swaps performed using two different methods:

1. `BPool.swapExactAmountIn` performs a direct token-to-token swap between two bound assets within the pool. Some amount `tokenAmountIn` of `tokenIn` is directly traded for some minimum amount `minAmountOut` of `tokenOut`. An

additional parameter, `maxPrice`, allows the trader to specify the maximum amount of slippage allowed during the trade.

2. `BPool.joinswapExternAmountIn` allows a trader to exchange an amount `tokenAmountIn` of `tokenIn` for a minimum amount `minPoolAmountOut` of the pool's token. A subsequent call to `BPool.exitswapPoolAmountIn` allows a trader to exchange amount `poolAmountIn` of the pool's tokens for a minimum amount `minAmountOut` of `tokenOut`.

While the latter method performs a swap by way of the pool's token as an intermediary, both methods can be used in order to perform a token-to-token swap. Our comparison between the two tested the relative amount `tokenAmountOut` of `tokenOut` between the two methods with a variety of different parameters.

Examples

Each example made use of a testing contract, found here: https://gist.github.com/wadeAlexC/12ee22438e8028f5439c5f0faaf9b7f7

Additionally, `BPool` was modified; unneeded functions were removed so that deployment did not exceed the block gas limit.

---

1. `tokenIn` weight: `25 BONE`

`tokenOut` weight: `25 BONE`

`tokenIn`, `tokenOut` at equal balances (`50 BONE`)

`tokenAmountIn`: `1 BONE`

`swapExactAmountIn` `tokenAmountOut`: 980391195693945000

`joinswapExternAmountIn` + `exitSwapPoolAmountIn` `tokenAmountOut`: 980391186207949598

Result: `swapExactAmountIn` gives 1.00000001x more tokens

---

2. `tokenIn` weight: `1 BONE`

`tokenOut` weight: `49 BONE`

`tokenIn`, `tokenOut` at equal balances (`50 BONE`)

`tokenAmountIn`: `1 BONE`

`swapExactAmountIn` `tokenAmountOut`: 20202659955287800

`joinswapExternAmountIn` + `exitSwapPoolAmountIn` `tokenAmountOut`: 20202659970818843

Result: `joinswap/exitswap` gives 1.00000001x more tokens

---

3. `tokenIn` weight: `25 BONE`

`tokenOut` weight: `25 BONE`

`tokenIn`, `tokenOut` at equal balances (`1 BONE`)

`tokenAmountIn`: `0.5 BONE`

`swapExactAmountIn` `tokenAmountOut`: 333333111111037037

`joinswapExternAmountIn` + `exitSwapPoolAmountIn` `tokenAmountOut`: 333333055579388951

Result: `swapExactAmountIn` gives 1.000000167x more tokens

---

4. `tokenIn` weight: `25 BONE`

`tokenOut` weight: `25 BONE`

`tokenIn`, `tokenOut` at equal balances (`30 BONE`)

`tokenAmountIn`: `15 BONE`

`swapExactAmountIn` `tokenAmountOut`: 9999993333331111110

`joinswapExternAmountIn` + `exitSwapPoolAmountIn` `tokenAmountOut`: 9999991667381668530

Result: `swapExactAmountIn` gives 1.000000167x more tokens

The final test raised the swap fee from `MIN_FEE` (0.0001%) to `MAX_FEE` (10%):

1. `tokenIn` weight: `25 BONE`

`tokenOut` weight: `25 BONE`

`tokenIn`, `tokenOut` at equal balances (`30 BONE`)

`tokenAmountIn`: `15 BONE`

`swapExactAmountIn` `tokenAmountOut`: 9310344827586206910

`joinswapExternAmountIn` + `exitSwapPoolAmountIn` `tokenAmountOut`: 9177966102628338740

Result: `swapExactAmountIn` gives 1.014423536x more tokens

Recommendation

Our final test showed that with equivalent balances and weights, raising the swap fee to 10% had a drastic effect on relative `tokenAmountOut` received, with `swapExactAmountIn` yielding >1.44% more tokens than the `joinswap/exitswap` method.

Reading through Balancer's provided documentation, our assumption was that these two swap methods were roughly equivalent. Discussion with Balancer clarified that the `joinswap/exitswap` method applied two swap fees: one for single asset deposit, and one for single asset withdrawal. With the minimum swap fee, this double application proved to have relatively little impact on the difference between the two methods. In fact, some parameters resulted in higher relative yield from the `joinswap/exitswap` method. With the maximum swap fee, the double application was distinctly noticeable.

Given the relative complexity of the math behind `BPool`s, there is much that remains to be tested. There are alternative swap methods, as well as numerous additional permutations of parameters that could be used; these tests were relatively narrow in scope.

We recommend increasing the intensity of unit testing to cover a more broad range of interactions with `BPool`'s various swap methods. In particular, the double application of the swap fee should be examined, as well as the differences between low and high swap fees.

Those using `BPool` should endeavor to understand as much of the underlying math as they can, ensuring awareness of the various options available for

performing trades.

5.2 Commented code exists in `BMath` Minor

Description

There are some instances of code being commented out in the `BMath.sol` that should be removed. It seems that most of the commented code is related to exit fee, however this is in contrast to `BPool.sol` code base that still has the exit fee code flow, but uses 0 as the fee.

Examples

code/contracts/BMath.sol:L137-L140

```
uint tokenInRatio = bdiv(newTokenBalanceIn, tokenBalanceIn);

// uint newPoolSupply = (ratioTi ^ weightTi) * poolSupply;
uint poolRatio = bpow(tokenInRatio, normalizedWeight);
```

code/contracts/BMath.sol:L206-L209

```
uint normalizedWeight = bdiv(tokenWeightOut, totalWeight);
// charge exit fee on the pool token side
// pAiAfterExitFee = pAi*(1-exitFee)
uint poolAmountInAfterExitFee = bmul(poolAmountIn, bsub(BONE, EXIT_FEE));
```

And many more examples.

Recommendation

Remove the commented code, or address them properly. If the code is related to exit fee, which is considered to be 0 in this version, this style should be persistent in other contracts as well.

5.3 Max weight requirement in `rebind` is inaccurate Minor

Description

`BPool.rebind` enforces `MIN_WEIGHT` and `MAX_WEIGHT` bounds on the passed-in `denorm` value:

code/contracts/BPool.sol:L262-L274

```
function rebind(address token, uint balance, uint denorm)
    public
    _logs_
    _lock_
{

    require(msg.sender == _controller, "ERR_NOT_CONTROLLER");
    require(_records[token].bound, "ERR_NOT_BOUND");
    require(!_finalized, "ERR_IS_FINALIZED");

    require(denorm >= MIN_WEIGHT, "ERR_MIN_WEIGHT");
    require(denorm <= MAX_WEIGHT, "ERR_MAX_WEIGHT");
    require(balance >= MIN_BALANCE, "ERR_MIN_BALANCE");
```

`MIN_WEIGHT` is `1 BONE`, and `MAX_WEIGHT` is `50 BONE`.

Though a token weight of `50 BONE` may make sense in a single-token system, `BPool` is intended to be used with two to eight tokens. The sum of the weights of all tokens must not be greater than `50 BONE`.

This implies that a weight of `50 BONE` for any single token is incorrect, given that at least one other token must be present.

Recommendation

`MAX_WEIGHT` for any single token should be `MAX_WEIGHT - MIN_WEIGHT`, or `49 BONE`.

5.4 Switch modifier order in `BPool` Minor

Description

`BPool` functions often use modifiers in the following order: `_logs_`, `_lock_`. Because `_lock_` is a reentrancy guard, it should take precedence over `_logs_`. See example:

```
pragma solidity ^0.5.0;
pragma experimental ABIEncoderV2;

contract Target {

    string[] arr;

    modifier a() {
        // sA1
        arr.push("sA1");
        _;
        // sA2
        arr.push("sA2");
    }

    modifier b() {
        // sB1
        arr.push("sB1");
        _;
        // sB2
        arr.push("sB2");
    }

    // sA1 -> sB1 -> func -> sB2 -> sA2
    function test() public a b {
        arr.push("func");
    }

    function get() public view returns (string[] memory) {
        return arr;
    }
}
```

Recommendation

Place `_lock_` before other modifiers; ensuring it is the very first and very last thing to run when a function is called.

6 Document Change Log

Version	Date	Description
1.0	2020-05-15	Initial report

Appendix 2 - Files in Scope

This audit covered the following files:

File Name	SHA-1 Hash

File Name	SHA-1 Hash
contracts/BFactory.sol	0d193312bc81d4b96c468ae51b6dd27550b8e5ae
contracts/BPool.sol	04450c7c1e9d861475cd1e1d673b992c810af756
contracts/BToken.sol	2447c07499a00d39a5aec76b68c6d5d58928d64d
contracts/BNum.sol	f679764be21d158411032bfad7f658210058c4ca
contracts/BConst.sol	459521a827d8302be1fd6c16b77721aea8ef24a1
contracts/BColor.sol	6fc688e13f12d4dbff1aa44de0e1203b1e1dbdd9
contracts/BMath.sol	c5cde402b16dd6ea0263ec626ae559de370a1ddb

Appendix 3 - Artifacts

This section contains some of the artifacts generated during our review by automated tools, the test suite, etc. If any issues or recommendations were identified by the output presented here, they have been addressed in the appropriate section above.

A.3.1 MythX

MythX is a security analysis API for Ethereum smart contracts. It performs multiple types of analysis, including fuzzing and symbolic execution, to detect many common vulnerability types. The tool was used for automated vulnerability discovery for all audited contracts and libraries. More details on MythX can be found at mythx.io.

Below is the raw output of the MythX vulnerability scan:

Report for /code/contracts/test/ttoken.sol

View on MythX Dashboard

No issues have been found.

Report for /code/contracts/test/tmath.sol

View on MythX Dashboard

High	Medium	Low	Unknown
1	0	0	0

- **Issue:** SWC-101 - Integer Overflow and Underflow
- **Severity:** High
- **Description:** It is possible to cause an integer overflow or underflow in the arithmetic operation.
- **Location:** /code/contracts/bnum.sol
- **Line:** 67
- **Column:** 18

```
{
    uint c0 = a * b;
    require(a == 0 || c0 / a == b, "ERR_MUL_OVERFLOW");
```

Report for /code/contracts/test/echidna/tbpooljoinpool.sol

View on MythX Dashboard

No issues have been found.

Report for /code/contracts/test/echidna/tbpooljoinexitpoolnofee.sol

View on MythX Dashboard

No issues have been found.

Report for /code/contracts/test/echidna/tbpooljoinexitpool.sol

View on MythX Dashboard

No issues have been found.

Report for /code/contracts/btoken.sol

View on MythX Dashboard

No issues have been found.

Report for /code/contracts/btoken.sol

View on MythX Dashboard

No issues have been found.

Report for /code/contracts/btoken.sol

View on MythX Dashboard

No issues have been found.

Report for /code/contracts/bpool.sol

View on MythX Dashboard

High	Medium	Low	Unknown
0	0	1	0

- **Issue:** SWC-123 - Requirement Violation
- **Severity:** Low
- **Description:** A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).
- **Location:** /code/contracts/bpool.sol
- **Line:** 711
- **Column:** 20

```
{
    bool xfer = IERC20(erc20).transfer(to, amount);
    require(xfer, "ERR_ERC20_FALSE");
```

- **Issue:** SWC-123 - Requirement Violation
- **Severity:** Low
- **Description:** A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).
- **Location:** /code/contracts/btoken.sol

- **Line:** 132
- **Column:** 8

```
function transferFrom(address src, address dst, uint amt) external returns (bool) {
    require(msg.sender == src || amt <= _allowance[src][msg.sender], "ERR_BTOKEN_BAD_
    _move(src, dst, amt);
```

Report for /code/contracts/bnum.sol

View on MythX Dashboard

No issues have been found.

Report for /code/contracts/bmath.sol

View on MythX Dashboard

No issues have been found.

Report for /code/contracts/bfactory.sol

View on MythX Dashboard

High	Medium	Low	Unknown
0	0	1	0

- **Issue:** SWC-123 - Requirement Violation
- **Severity:** Low
- **Description:** A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).
- **Location:** /code/contracts/bfactory.sol
- **Line:** 75
- **Column:** 25

```
    require(msg.sender == _blabs, "ERR_NOT_BLABS");
    uint collected = IERC20(pool).balanceOf(address(this));
    bool xfer = pool.transfer(_blabs, collected); //@audit-info fails if not bool ret
```

- **Issue:** SWC-123 - Requirement Violation

- **Severity:** Low
- **Description:** A requirement was violated in a nested call and the call was reverted as a result. Make sure valid inputs are provided to the nested call (for instance, via passed arguments).
- **Location:** /code/contracts/bfactory.sol
- **Line:** 20
- **Column:** 0

```
contract BFactory is BBronze {//@audit-ok checked, has minor stuff
event LOG_NEW_POOL(
```

Report for /code/contracts/bconst.sol

View on MythX Dashboard

No issues have been found.

Report for /code/contracts/bcolor.sol

View on MythX Dashboard

No issues have been found.

Report for /code/contracts/bcolor.sol

View on MythX Dashboard

No issues have been found.

A.3.2 Ethlint

Ethlint is an open source project for linting Solidity code. Only security-related issues were reviewed by the audit team.

Below is the raw output of the Ethlint vulnerability scan:

```
contracts/BColor.sol
  26:12    error    Only use indent of 8 spaces.    indentation
  27:0     error    Only use indent of 4 spaces.    indentation

contracts/BConst.sol
```

```
  40:1    warning    Line contains trailing whitespace    no-trailing-whites

contracts/BMath.sol
  28:1     warning    Line contains trailing whitespace    no-trailing-white
  42:1     warning    Line contains trailing whitespace    no-trailing-white
  133:1    warning    Line contains trailing whitespace    no-trailing-white
  170:6    warning    Line contains trailing whitespace    no-trailing-white
  172:1    warning    Line contains trailing whitespace    no-trailing-white
  176:1    warning    Line contains trailing whitespace    no-trailing-white
  212:5    warning    Line contains trailing whitespace    no-trailing-white
  219:1    warning    Line contains trailing whitespace    no-trailing-white
  221:1    warning    Line contains trailing whitespace    no-trailing-white
  249:1    warning    Line contains trailing whitespace    no-trailing-white
  253:1    warning    Line contains trailing whitespace    no-trailing-white

contracts/BNum.sol
  21:1     warning    Line contains trailing whitespace
  89:4     error      "bpowi": Avoid assigning to function parameters.
  115:3    warning    Line contains trailing whitespace
  115:8    warning    Assignment operator must have exactly single space on
  133:8    warning    Assignment operator must have exactly single space on
  134:8    warning    Assignment operator must have exactly single space on
  136:8    warning    Assignment operator must have exactly single space on
  140:1    warning    Line contains trailing whitespace

contracts/BPool.sol
  66:1      warning    Line contains trailing whitespace
  117:1     warning    Line contains trailing whitespace
  144:1     warning    Line contains trailing whitespace
  144:8     warning    Line exceeds the limit of 145 characters
  173:1     warning    Line contains trailing whitespace
  197:1     warning    Line contains trailing whitespace
  214:1     warning    Line contains trailing whitespace
  255:21    warning    "0" should be immediately followed by a comma, then a
  282:1     warning    Line contains trailing whitespace
  283:8     warning    Line contains trailing whitespace
  334:1     warning    Line contains trailing whitespace
  383:12    warning    Line exceeds the limit of 145 characters
  401:1     warning    Line contains trailing whitespace
  443:8     warning    Line exceeds the limit of 145 characters
  446:36    error      Only use indent of 12 spaces.
  447:36    error      Only use indent of 12 spaces.
  448:36    error      Only use indent of 12 spaces.
  449:36    error      Only use indent of 12 spaces.
  450:36    error      Only use indent of 12 spaces.
  451:0     error      Only use indent of 8 spaces.
  455:28    error      Only use indent of 12 spaces.
  456:28    error      Only use indent of 12 spaces.
  457:28    error      Only use indent of 12 spaces.
  458:28    error      Only use indent of 12 spaces.
  459:28    error      Only use indent of 12 spaces.
  460:28    error      Only use indent of 12 spaces.
```

```
    461:0     error      Only use indent of 8 spaces.
    468:32    error      Only use indent of 12 spaces.
    469:32    error      Only use indent of 12 spaces.
    470:32    error      Only use indent of 12 spaces.
    471:32    error      Only use indent of 12 spaces.
    472:32    error      Only use indent of 12 spaces.
    473:0     error      Only use indent of 8 spaces.
    476:1     warning    Line contains trailing whitespace
    495:1     warning    Line contains trailing whitespace
    508:36    error      Only use indent of 12 spaces.
    509:36    error      Only use indent of 12 spaces.
    510:36    error      Only use indent of 12 spaces.
    511:36    error      Only use indent of 12 spaces.
    512:36    error      Only use indent of 12 spaces.
    513:0     error      Only use indent of 8 spaces.
    517:28    error      Only use indent of 12 spaces.
    518:28    error      Only use indent of 12 spaces.
    519:28    error      Only use indent of 12 spaces.
    520:28    error      Only use indent of 12 spaces.
    521:28    error      Only use indent of 12 spaces.
    522:28    error      Only use indent of 12 spaces.
    523:0     error      Only use indent of 8 spaces.
    530:32    error      Only use indent of 12 spaces.
    531:32    error      Only use indent of 12 spaces.
    532:32    error      Only use indent of 12 spaces.
    533:32    error      Only use indent of 12 spaces.
    534:32    error      Only use indent of 12 spaces.
    535:0     error      Only use indent of 8 spaces.
    555:8     warning    Line contains trailing whitespace
    563:28    error      Only use indent of 12 spaces.
    564:28    error      Only use indent of 12 spaces.
    565:28    error      Only use indent of 12 spaces.
    566:28    error      Only use indent of 12 spaces.
    567:28    error      Only use indent of 12 spaces.
    568:28    error      Only use indent of 12 spaces.
    569:0     error      Only use indent of 8 spaces.
    596:28    error      Only use indent of 12 spaces.
    597:28    error      Only use indent of 12 spaces.
    598:28    error      Only use indent of 12 spaces.
    599:28    error      Only use indent of 12 spaces.
    600:28    error      Only use indent of 12 spaces.
    601:28    error      Only use indent of 12 spaces.
    602:0     error      Only use indent of 8 spaces.
    606:8     warning    Line contains trailing whitespace
    632:28    error      Only use indent of 12 spaces.
    633:28    error      Only use indent of 12 spaces.
    634:28    error      Only use indent of 12 spaces.
    635:28    error      Only use indent of 12 spaces.
    636:28    error      Only use indent of 12 spaces.
    637:28    error      Only use indent of 12 spaces.
    638:0     error      Only use indent of 8 spaces.
    641:8     warning    Line contains trailing whitespace
    671:28    error      Only use indent of 12 spaces.
```

```
  671:28    error      Only use indent of 12 spaces.
  672:28    error      Only use indent of 12 spaces.
  673:28    error      Only use indent of 12 spaces.
  674:28    error      Only use indent of 12 spaces.
  675:28    error      Only use indent of 12 spaces.
  676:28    error      Only use indent of 12 spaces.
  677:0     error      Only use indent of 8 spaces.
  691:8     warning    Line contains trailing whitespace

contracts/test/TToken.sol
  41:8    warning    Provide an error message for require()    error-reason
  44:8    warning    Provide an error message for require()    error-reason

contracts/test/echidna/TBPoolJoinExitPool.sol
  3:0     warning    "pragma solidity 0.5.12;" should be at the top of the f
  41:4    error      "joinAndExitPool": Avoid assigning to function paramete
  41:4    error      "joinAndExitPool": Avoid assigning to function paramete
  46:8    warning    Provide an error message for require()
  47:8    warning    Provide an error message for require()
  48:8    warning    Provide an error message for require()
  49:8    warning    Provide an error message for require()
  54:1    warning    Line contains trailing whitespace
  54:8    warning    Provide an error message for require()
  56:8    warning    Provide an error message for require()
  58:8    warning    Provide an error message for require()
  60:1    warning    Line contains trailing whitespace
  61:1    warning    Line contains trailing whitespace
  61:8    warning    Provide an error message for require()
  62:1    warning    Line contains trailing whitespace
  62:8    warning    Provide an error message for require()

contracts/test/echidna/TBPoolJoinExitPoolNoFee.sol
  3:0     warning    "pragma solidity 0.5.12;" should be at the top of the f
  38:4    error      "joinAndExitNoFeePool": Avoid assigning to function par
  38:4    error      "joinAndExitNoFeePool": Avoid assigning to function par
  39:1    warning    Line contains trailing whitespace
  45:8    warning    Provide an error message for require()
  46:8    warning    Provide an error message for require()
  47:8    warning    Provide an error message for require()
  48:8    warning    Provide an error message for require()
  53:1    warning    Line contains trailing whitespace
  53:8    warning    Provide an error message for require()
  55:8    warning    Provide an error message for require()
  57:8    warning    Provide an error message for require()
  59:1    warning    Line contains trailing whitespace
  60:1    warning    Line contains trailing whitespace
  60:8    warning    Provide an error message for require()
  61:1    warning    Line contains trailing whitespace
  61:8    warning    Provide an error message for require()

contracts/test/echidna/TBPoolJoinPool.sol
  3:0     warning    "pragma solidity 0.5.12;" should be at the top of the f
  17:8    warning    Provide an error message for require()
```

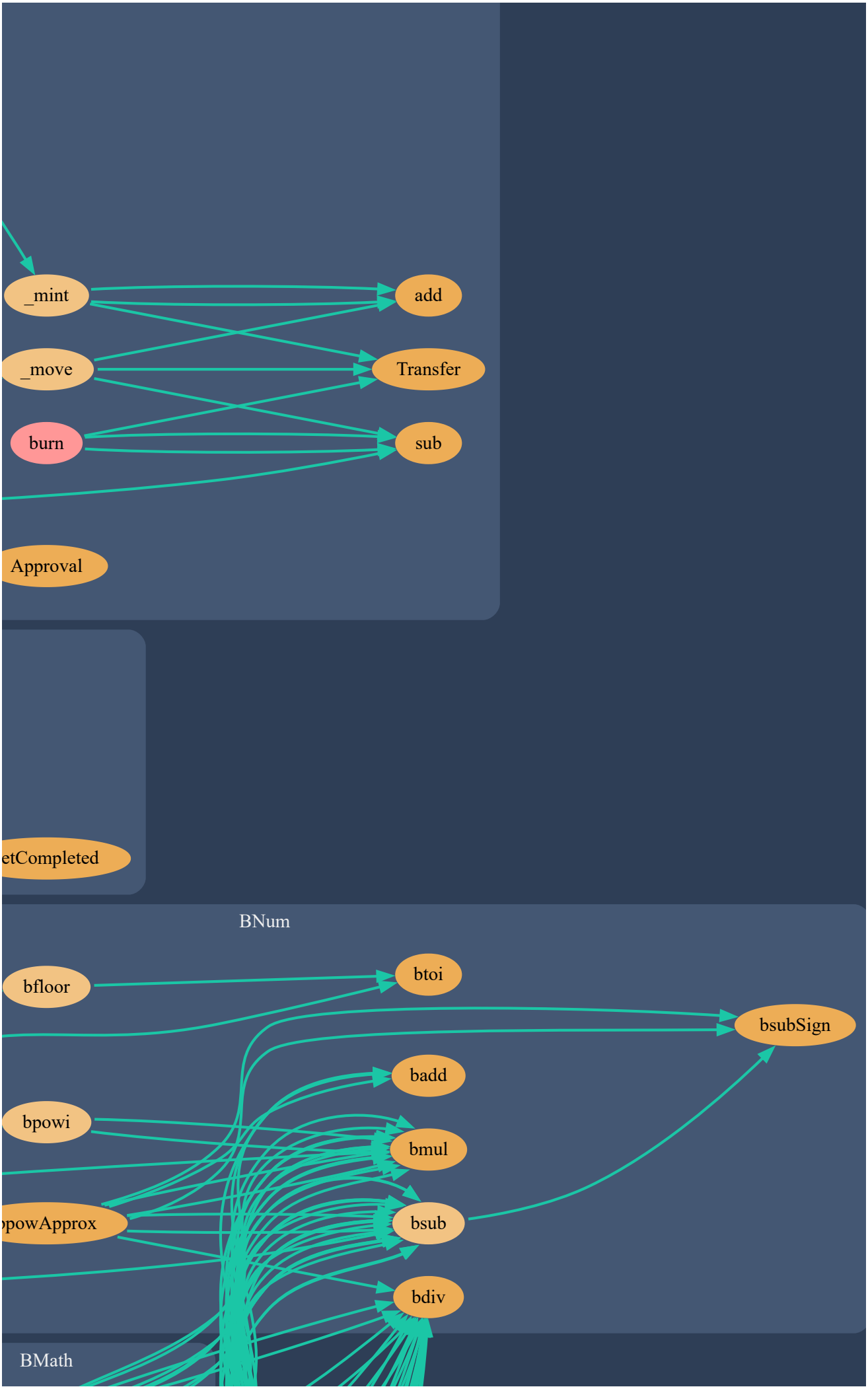
```
  18:8    warning    Provide an error message for require()
  19:8    warning    Provide an error message for require()
  20:8    warning    Provide an error message for require()
  28:8    warning    Provide an error message for require()
  29:8    warning    Provide an error message for require()

✖ 73 errors, 77 warnings found.
```

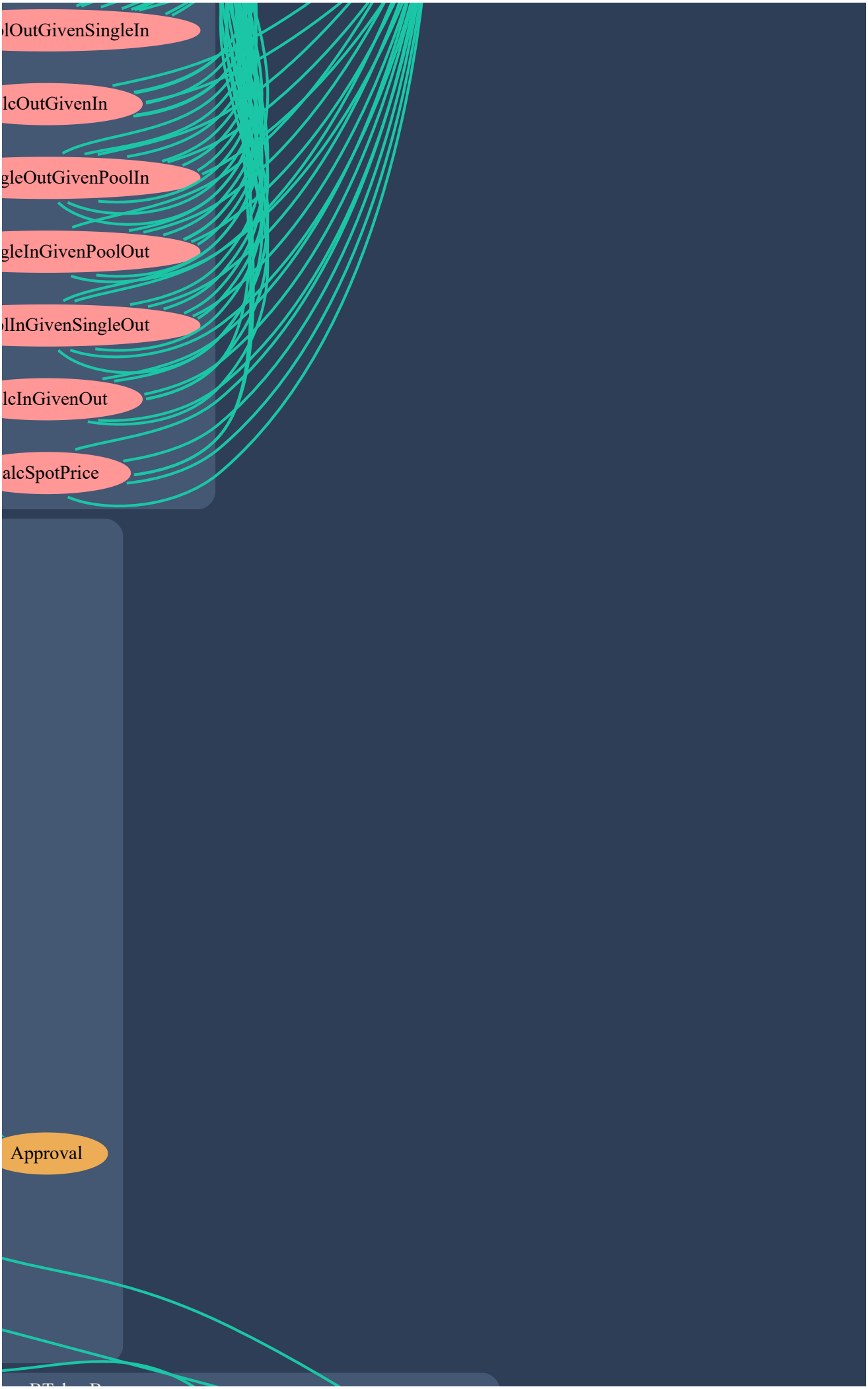
A.3.3 Surya

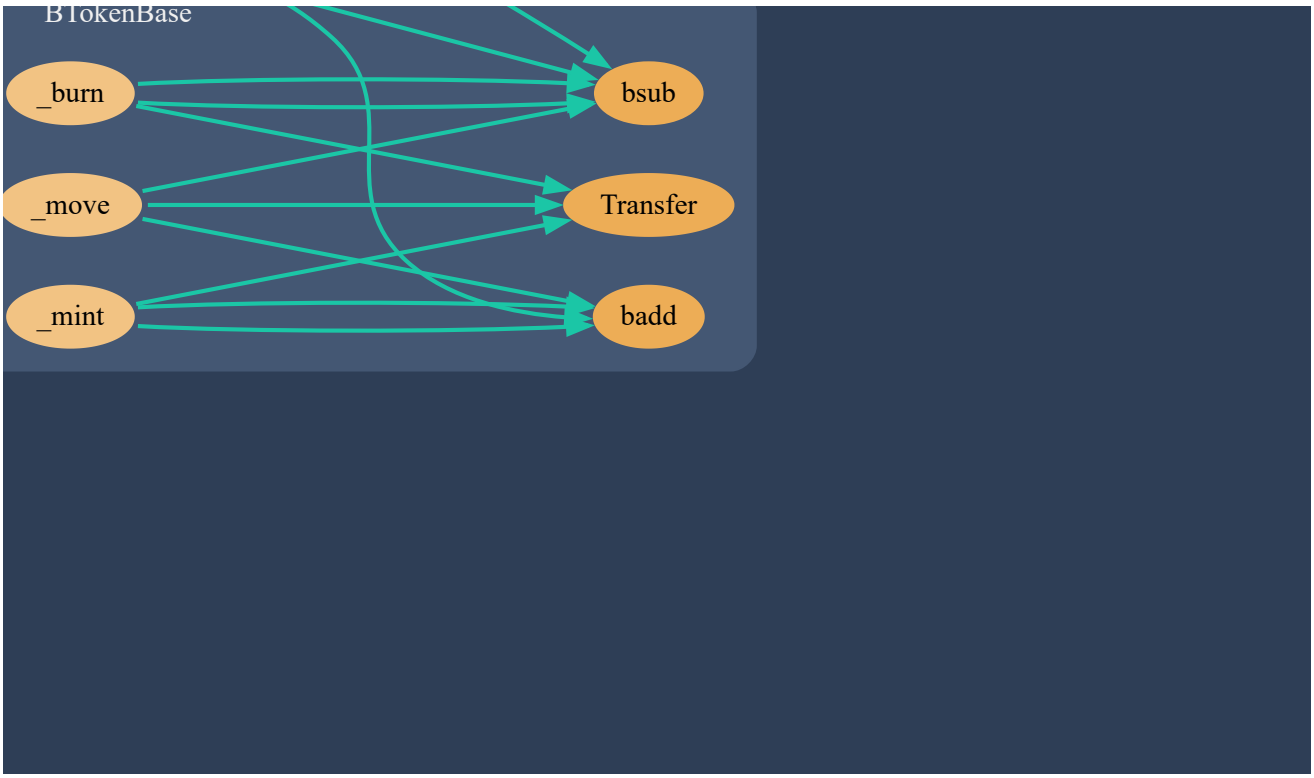
Surya is a utility tool for smart contract systems. It provides a number of visual outputs and information about the structure of smart contracts. It also supports querying the function call graph in multiple ways to aid in the manual inspection and control flow analysis of contracts.

Below is a complete list of functions with their visibility and modifiers:

File Name	SHA-1 Hash
contracts/BPool.sol	04450c7c1e9d861475cd1e1d673b992c810af756
contracts/BToken.sol	2447c07499a00d39a5aec76b68c6d5d58928d64d
contracts/BNum.sol	f679764be21d158411032bfad7f658210058c4ca
contracts/BConst.sol	459521a827d8302be1fd6c16b77721aea8ef24a1
contracts/BColor.sol	6fc688e13f12d4dbff1aa44de0e1203b1e1dbdd9
contracts/BMath.sol	c5cde402b16dd6ea0263ec626ae559de370a1ddb


_mint
add
_move
Transfer
burn
sub
Approval
etCompleted
BNum
bfloor
btoi
bsubSign
badd
bpowi
bmul
powApprox
bsub
bdiv
BMath


lOutGivenSingleIn
lcOutGivenIn
gleOutGivenPoolIn
gleInGivenPoolOut
lInGivenSingleOut
lcInGivenOut
alcSpotPrice
Approval



Contract	Type	Bases		
L	**Function Name**	**Visibility**	**Mutability**	**Modifiers**
BPool	Implementation	BBronze, BToken, BMath		
L		Public ❗	🛑	NO❗
L	isPublicSwap	External ❗		NO❗
L	isFinalized	External ❗		NO❗
L	isBound	External ❗		NO❗
L	getNumTokens	External ❗		NO❗
L	getCurrentTokens	External ❗		*viewlock*
L	getFinalTokens	External ❗		*viewlock*
L	getDenormalizedWeight	External ❗		*viewlock*

Contract	Type	Bases		
L	getTotalDenormalizedWeight	External !		*viewlock*
L	getNormalizedWeight	External !		*viewlock*
L	getBalance	External !		*viewlock*
L	getSwapFee	External !		*viewlock*
L	getController	External !		*viewlock*
L	setSwapFee	External !	●	*logs lock*
L	setController	External !	●	*logs lock*
L	setPublicSwap	External !	●	*logs lock*
L	finalize	External !	●	*logs lock*
L	bind	External !	●	*logs*
L	rebind	Public !	●	*logs lock*
L	unbind	External !	●	*logs lock*
L	gulp	External !	●	*logs lock*
L	getSpotPrice	External !		*viewlock*
L	getSpotPriceSansFee	External !		*viewlock*
L	joinPool	External !	●	*logs lock*
L	exitPool	External !	●	*logs lock*
L	swapExactAmountIn	External !	●	*logs lock*
L	swapExactAmountOut	External !	●	*logs lock*
L	joinswapExternAmountIn	External !	●	*logs lock*

Contract	Type	Bases		
L	joinswapPoolAmountOut	External ❗	🛑	*logs lock*
L	exitswapPoolAmountIn	External ❗	🛑	*logs lock*
L	exitswapExternAmountOut	External ❗	🛑	*logs lock*
L	_pullUnderlying	Internal 🔒	🛑	
L	_pushUnderlying	Internal 🔒	🛑	
L	_pullPoolShare	Internal 🔒	🛑	
L	_pushPoolShare	Internal 🔒	🛑	
L	_mintPoolShare	Internal 🔒	🛑	
L	_burnPoolShare	Internal 🔒	🛑	
IERC20	Interface			
L	totalSupply	External ❗		NO❗
L	balanceOf	External ❗		NO❗
L	allowance	External ❗		NO❗
L	approve	External ❗	🛑	NO❗
L	transfer	External ❗	🛑	NO❗
L	transferFrom	External ❗	🛑	NO❗
BTokenBase	Implementation	BNum		
L	_mint	Internal 🔒	🛑	

Contract	Type	Bases		
└	_burn	Internal 🔒	🛑	
└	_move	Internal 🔒	🛑	
└	_push	Internal 🔒	🛑	
└	_pull	Internal 🔒	🛑	
BToken	Implementation	BTokenBase, IERC20		
└	name	Public ❗		NO❗
└	symbol	Public ❗		NO❗
└	decimals	Public ❗		NO❗
└	allowance	External ❗		NO❗
└	balanceOf	External ❗		NO❗
└	totalSupply	Public ❗		NO❗
└	approve	External ❗	🛑	NO❗
└	increaseApproval	External ❗	🛑	NO❗
└	decreaseApproval	External ❗	🛑	NO❗
└	transfer	External ❗	🛑	NO❗
└	transferFrom	External ❗	🛑	NO❗
BNum	Implementation	BConst		
└	btoi	Internal 🔒		
└	bfloor	Internal 🔒		
└	badd	Internal 🔒		
└	bsub	Internal 🔒		

Contract	Type	Bases		
L	bsubSign	Internal 🔒		
L	bmul	Internal 🔒		
L	bdiv	Internal 🔒		
L	bpowi	Internal 🔒		
L	bpow	Internal 🔒		
L	bpowApprox	Internal 🔒		
BConst	Implementation	BBronze		
BColor	Implementation			
L	getColor	External ❗		NO❗
BBronze	Implementation	BColor		
L	getColor	External ❗		NO❗
BMath	Implementation	BBronze, BConst, BNum		
L	calcSpotPrice	Public ❗		NO❗
L	calcOutGivenIn	Public ❗		NO❗
L	calcInGivenOut	Public ❗		NO❗
L	calcPoolOutGivenSingleIn	Public ❗		NO❗
L	calcSingleInGivenPoolOut	Public ❗		NO❗

Contract	Type	Bases		
L	calcSingleOutGivenPoolIn	Public ❗		NO❗
L	calcPoolInGivenSingleOut	Public ❗		NO❗

Legend

Symbol	Meaning
🛑	Function can modify state
💵	Function is payable

A.3.4 Tests Suite

Below is the output generated by running the test suite:

```
➜ 💀  code (master) ✗ yarn test:verbose
yarn run v1.22.4
$ VERBOSE=true truffle test
Using network 'development'.


Compiling your contracts...
===========================
> Everything is up to date, there is nothing to compile.



  Contract: BFactory
    Factory
      ✓ BFactory is bronze release
      ✓ isBPool on non pool returns false
      ✓ isBPool on pool returns true
      ✓ fails nonAdmin calls collect (55ms)
      ✓ admin collects fees (586ms)
      ✓ nonadmin cant set blabs address (40ms)
      ✓ admin changes blabs address (55ms)

  Contract: BPool
    Extreme weights
output[0]
expected: 8.23390841016124456)
actual  : 8.233908370260792)
```

```
relDif  : 4.845870341569494063 5e-9)
output[1]
expected: 74.1844011380065814)
actual  : 74.184401135022015545)
relDif  : 4.0231717304662987451e-11)
      ✓ swapExactAmountIn (225ms)
output[0]
expected: 425506505648.348073)
actual  : 425506505648.348072674947244244)
relDif  : 7.6391959098419471932e-19)
output[1]
expected: 31306034272.9265099)
actual  : 31306034272.926509852164468306)
relDif  : 1.5279971674779713695e-18)
      ✓ swapExactAmountOut (109ms)
Pool Balance
expected: 101)
actual  : 101)
relDif  : 0)
WETH Balance
expected: 1010)
actual  : 1010)
relDif  : 0)
Dai Balance
expected: 1010)
actual  : 1010)
relDif  : 0)
      ✓ joinPool (225ms)
Pool Balance
expected: 100)
actual  : 100)
relDif  : 0)
WETH Balance
expected: 1000)
actual  : 999.99999999999999999)
relDif  : 1e-20)
Dai Balance
expected: 1000)
actual  : 999.99999999999999999)
relDif  : 1e-20)
      ✓ exitPool (177ms)
Pool Balance
expected: 100.1908021557112462)
actual  : 100.1908021555181693)
relDif  : 1.9270920667940166078e-12)
WETH Balance
expected: 1100.0980961342116)
actual  : 1100.09809613421159999)
relDif  : 9.0900984513475635928e-21)
Dai Balance
expected: 1000)
actual  : 999.99999999999999999)
```

```
relDif  : 1e-20)
      ✓ joinswapExternAmountIn (198ms)
Pool Balance
expected: 110.20988237128237082)
actual  : 110.2098823710893023)
relDif  : 1.7518258421652057304e-12)
WETH Balance
expected: 1100.0980961342116)
actual  : 1100.09809613421159999)
relDif  : 9.0900984513475635928e-21)
Dai Balance
expected: 1102.1437413959127689)
actual  : 1102.14374139394507189)
relDif  : 1.7853361009951628812e-12)
      ✓ joinswapPoolAmountOut (191ms)
      ✓ joinswapExternAmountIn should revert (53ms)
      ✓ joinswapPoolAmountOut should revert (2036ms)
      ✓ exitswapExternAmountOut should revert (49ms)
      ✓ exitswapPoolAmountIn should revert (116ms)
Pool Balance
expected: 99.188894134154133738)
actual  : 99.188894134473583336)
relDif  : 3.2206186064333039216e-12)
WETH Balance
expected: 1100.0980961342116)
actual  : 1100.09809613421159999)
relDif  : 9.0900984513475635928e-21)
Dai Balance
expected: 989.8010445541475889)
actual  : 989.80104455217989189)
relDif  : 1.9879722504095176229e-12)
      ✓ exitswapExternAmountOut (201ms)
tokenAmountIn: 56902575375739370966)
poolAmountOut
expected: 0.1)
actual  : 0.099999999746038493)
relDif  : 2.53961507e-9)
      ✓ poolAmountOut = joinswapExternAmountIn(joinswapPoolAmountOut(poolAm
poolAmountOut: 98203766296104227)
tokenAmountIn
expected: 1)
actual  : 1.00000000000644237)
relDif  : 6.44237e-12)
      ✓ tokenAmountIn = joinswapPoolAmountOut(joinswapExternAmountIn(tokenA
tokenAmountOut: 54053762074497907547)
poolAmountIn
expected: 0.1)
actual  : 0.099999999803759585)
relDif  : 1.96240415e-9)
      ✓ poolAmountIn = exitswapExternAmountOut(exitswapPoolAmountIn(poolAmo
poolAmountIn: 98209678977295947)
tokenAmountOut
expected: 1)
```

```
expected: 1)
actual  : 0.999999999993555321)
relDif  : 6.444679e-12)
      ✓ tokenAmountOut = exitswapPoolAmountIn(exitswapExternAmountOut(token/

  Contract: BPool
    With fees
output[0]
expected: 3.9973324441480493498)
actual  : 3.997332444148049352)
relDif  : 5.503670337003670337e-19)
output[1]
expected: 0.75050050050050050071)
actual  : 0.7505005005005005)
relDif  : 9.4603534511503834585e-19)
      ✓ swapExactAmountIn (87ms)
output[0]
expected: 4.0040040040040040036)
actual  : 4.004004004004004)
relDif  : 8.9910000000000000009e-19)
output[1]
expected: 5.340008009344012012)
actual  : 5.340008009344012012)
relDif  : 0)
      ✓ swapExactAmountOut (89ms)
Pool Balance
expected: 101)
actual  : 101)
relDif  : 0)
WETH Balance
expected: 4.04)
actual  : 4.04)
relDif  : 0)
Dai Balance
expected: 12.12)
actual  : 12.12)
relDif  : 0)
      ✓ joinPool (204ms)
Pool Balance
expected: 100)
actual  : 100)
relDif  : 0)
WETH Balance
expected: 4)
actual  : 4)
relDif  : 0)
Dai Balance
expected: 12)
actual  : 12)
relDif  : 0)
      ✓ exitPool (162ms)
pAo
expected: 10)
```

```
expected: 10)
actual  : 9.9999999991934343)
relDif  : 8.065657e-11)
Pool Balance
expected: 110)
actual  : 109.9999999991934343)
relDif  : 7.3324154545454545455e-12)
WETH Balance
expected: 4.8404202101050532)
actual  : 4.8404202101050532)
relDif  : 0)
Dai Balance
expected: 12)
actual  : 12)
relDif  : 0)
      ✓ joinswapExternAmountIn (367ms)
tAi
expected: 2.52126063031516)
actual  : 2.521260630334527224)
relDif  : 7.681563646031738655e-12)
Pool Balance
expected: 121)
actual  : 120.9999999991934443)
relDif  : 6.6657495867768595041e-12)
WETH Balance
expected: 4.8404202101050532)
actual  : 4.8404202101050532)
relDif  : 0)
Dai Balance
expected: 14.52126063031516)
actual  : 14.521260630334527224)
relDif  : 1.3337150604933165752e-12)
      ✓ joinswapPoolAmountOut (224ms)
tAo
expected: 0.9192199999999999343)
actual  : 0.91922000000580478)
relDif  : 6.3149688866647814613e-12)
Pool Balance
expected: 108.900000000000002)
actual  : 108.8999999991934463)
relDif  : 7.4063884297520659797e-12)
WETH Balance
expected: 3.9212002101050532657)
actual  : 3.92120021009924842)
relDif  : 1.4803747294108407181e-12)
Dai Balance
expected: 14.52126063031516)
actual  : 14.521260630334527224)
relDif  : 1.3337150604933165752e-12)
      ✓ exitswapPoolAmountIn (234ms)
pAi
expected: 10.8900000000000002)
actual  : 10.889999997872901711)
```

```
relDif  : 1.9532584839302111671e-10)
Pool Balance
expected: 98.0100000000000018)
actual  : 98.010000001320544589)
relDif  : 1.3473551566166717434e-11)
WETH Balance
expected: 3.9212002101050532657)
actual  : 3.92120021009924842)
relDif  : 1.4803747294108407181e-12)
Dai Balance
expected: 11.76360063031516)
actual  : 11.763600630334527224)
relDif  : 1.6463687104516340961e-12)
      ✓ exitswapExternAmountOut (399ms)
tAi: 841404486126606882)
pAo
expected: 10)
actual  : 9.999999998963901476)
relDif  : 1.036098524e-10)
      ✓ pAo = joinswapExternAmountIn(joinswapPoolAmountOut(pAo)) (238ms)
pAo: 4078858999812498739)
tAi
expected: 1)
actual  : 0.999999999942422639)
relDif  : 5.7577361e-11)
      ✓ tAi = joinswapPoolAmountOut(joinswapExternAmountIn(tAi)) (197ms)
tAo: 758963127737565681)
pAi
expected: 10)
actual  : 9.999999999604227562)
relDif  : 3.95772438e-11)
      ✓ pAi = exitswapExternAmountOut(exitswapPoolAmountIn(pAi)) (258ms)
pAi: 4260502505087206679)
tAo
expected: 1)
actual  : 0.999999999938397128)
relDif  : 6.1602872e-11)
      ✓ tAo = exitswapPoolAmountIn(exitswapExternAmountOut(tAo)) (214ms)

  Contract: TMath
    BMath
      ✓ badd throws on overflow
      ✓ bsub throws on underflow
      ✓ bmul throws on overflow
      ✓ bdiv throws on div by 0
      ✓ bpow throws on base outside range

  Contract: BPool
    Binding Tokens
      ✓ Admin approves tokens (307ms)
      ✓ Admin binds tokens (544ms)
      ✓ Fails binding more than 8 tokens (40ms)
```

```
      ✓ Rebind token at a smaller balance (139ms)
      ✓ Fails gulp on unbound token (46ms)
      ✓ Pool can gulp tokens (100ms)
      ✓ Fails swapExactAmountIn with limits (471ms)
      ✓ Fails swapExactAmountOut with limits (347ms)

  Contract: BPool
    Binding Tokens
      ✓ Controller is msg.sender
      ✓ Pool starts with no bound tokens (39ms)
      ✓ Fails binding tokens that are not approved (84ms)
      ✓ Admin approves tokens (132ms)
      ✓ Fails binding weights and balances outside MIX MAX (270ms)
      ✓ Fails finalizing pool without 2 tokens (43ms)
      ✓ Admin binds tokens (284ms)
      ✓ Admin unbinds token (216ms)
      ✓ Fails binding above MAX TOTAL WEIGHT (75ms)
      ✓ Fails rebinding token or unbinding random token (139ms)
      ✓ Get current tokens
      ✓ Fails getting final tokens before finalized
    Finalizing pool
      ✓ Fails when other users interact before finalizing (219ms)
      ✓ Fails calling any swap before finalizing (197ms)
      ✓ Fails calling any join exit swap before finalizing (181ms)
      ✓ Only controller can setPublicSwap (91ms)
      ✓ Fails setting low swap fees (58ms)
      ✓ Fails setting high swap fees (42ms)
      ✓ Fails nonadmin sets fees or controller (97ms)
      ✓ Admin sets swap fees (63ms)
      ✓ Fails nonadmin finalizes pool (49ms)
      ✓ Admin finalizes pool (90ms)
      ✓ Fails finalizing pool after finalized (49ms)
      ✓ Cant setPublicSwap, setSwapFee when finalized (90ms)
      ✓ Fails binding new token after finalized (94ms)
      ✓ Fails unbinding after finalized (46ms)
      ✓ Get final tokens
    User interactions
      ✓ Other users approve tokens (278ms)
      ✓ User1 joins pool (147ms)
      ✓ Fails admin unbinding token after finalized and others joined (43ms)
      ✓ getSpotPriceSansFee and getSpotPrice (54ms)
      ✓ Fail swapExactAmountIn unbound or over min max ratios (95ms)
swapExactAmountIn
expected: 475.90580533709142153)
actual  : 475.905805337091423)
relDif  : 3.0888465396188565699e-18)
      ✓ swapExactAmountIn (160ms)
swapExactAmountOut
expected: 2.7582748244734202608)
actual  : 2.758274824473420261)
relDif  : 7.2509090909090909909e-20)
      ✓ swapExactAmountOut (104ms)
      ✓ Fails joins exits with limits (734ms)
```

```
      ✓ Fails joins exits with limits (734ms)
      ✓ Fails calling any swap on unbound token (384ms)
      ✓ Fails calling weights, balances, spot prices on unbound token (151ms
    BToken interactions
      ✓ Token descriptors (55ms)
      ✓ Token allowances (204ms)
      ✓ Token transfers (160ms)


  87 passing (20s)

✨  Done in 24.53s.
```

Appendix 4 - 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 website. 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.