FOAM Token Controller Audit

1 Summary

From August 24th, 2018 to August 31st, 2018, ConsenSys Diligence conducted a security audit on the FOAM token controller. The findings and recommendations are presented here in this report.

1.1 Audit Dashboard

Audit Details

- **Project Name:** FOAM Token Controller
- **Client Name:** Token Foundry
- **Client Contact:** Stuart Hunter
- **Auditors:** Steve Marx, John Mardlin
- **GitHub :** https://github.com/ConsenSys/foam-controller-audit-2018-08-24
- **Languages:** Solidity
- **Date:** 2018-08-24 to 2018-08-31

Number of issues per severity

	Minor	Medium	Major	Critical
Open	3	2	2	0
Closed	0	0	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

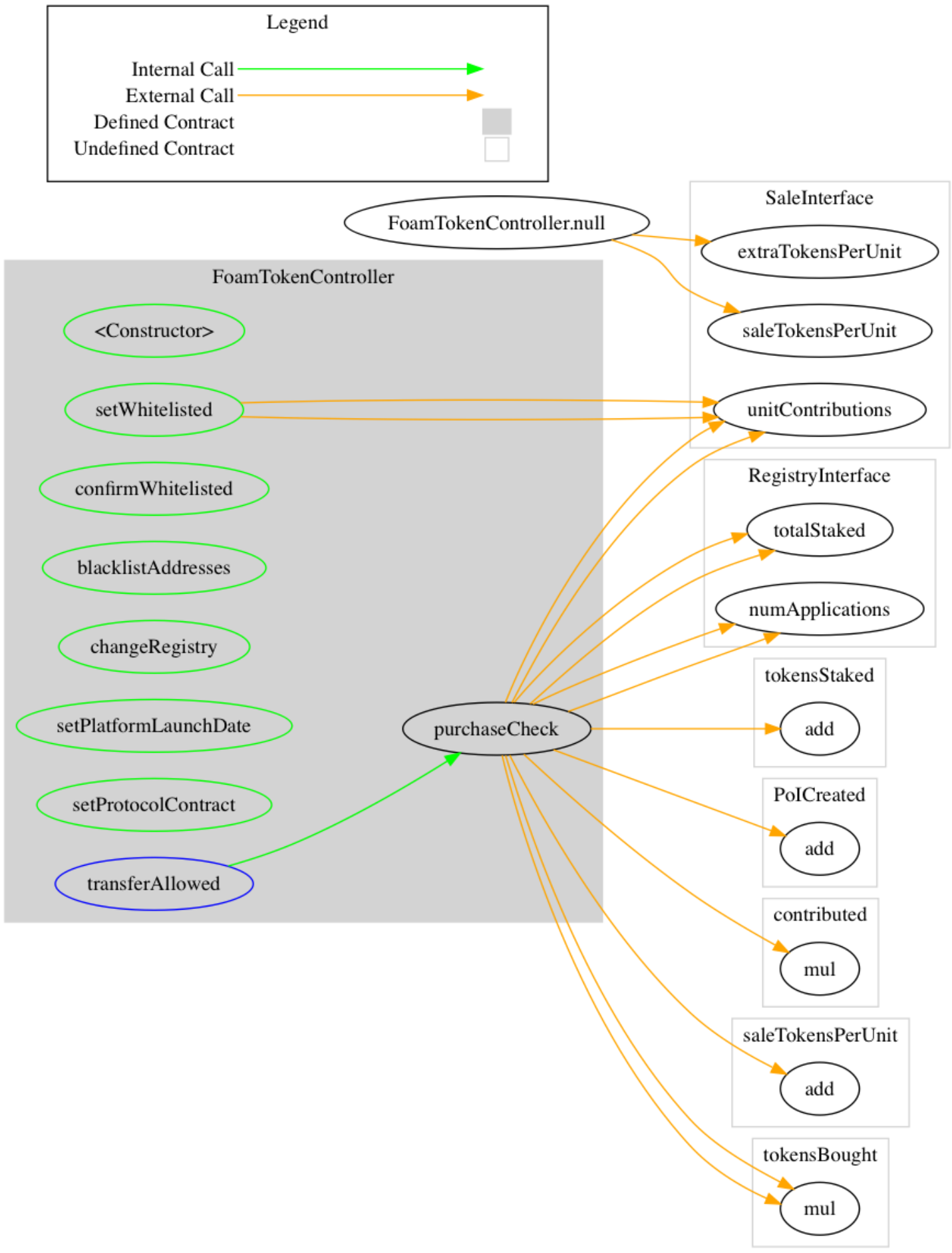
Documentation

The following documentation was available to the audit team:

- The Foam Token Controller spec explains the goals and basic design of the contract.

Scope

The audit focus was on the `FoamTokenController` contract. The following graph shows how this contract interacts with the token sale and FOAM registry contracts. Those contracts were considered out of scope for this audit.


Legend
Internal Call
External Call
Defined Contract
Undefined Contract
FoamTokenController.null
SaleInterface
extraTokensPerUnit
saleTokensPerUnit
unitContributions
FoamTokenController
<Constructor>
setWhitelisted
confirmWhitelisted
blacklistAddresses
changeRegistry
setPlatformLaunchDate
setProtocolContract
transferAllowed
purchaseCheck
RegistryInterface
totalStaked
numApplications
tokensStaked
add
PoICreated
add
contributed
mul
saleTokensPerUnit
add
tokensBought
mul

Flow graph

Design

The token controller contract restricts the ability for FOAM token holders to transfer tokens. Specifically, it requires token holders to complete "proof of use" before freely exchanging tokens.

1.4 Key Observations/Recommendations

- The contract design is fairly straightforward and closely follows the behavior described in the spec.
- **Test coverage is incomplete:** Any contract system that is used on the main net should have as a minimum requirement a 100% test coverage.
- **Fix all issues:** It is recommended to fix all the issues listed in the below chapters, at the very least the ones with severity Critical, Major and Medium. All issues have also been created as issues in a separate audit repository.
- **Comment the code:** There are a few places in the code where comments explaining the logic would make it easier to understand the code.

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 2. The table also contains the GitHub status of any discovered issue.

Chapter	Issue Title	Issue Status	Severity
3.1	When checking proof of use, an unconfirmed whitelisted pair is used.	Open	Major
3.2	Whitelisting mechanism allows immediate trading, without proof of use.	Open	Major
3.3	Blacklist feature is untested	Open	Medium
3.4	Whitelisting logic should be simplified.	Open	Medium
3.5	Token controller contract assumes behavior of other external contracts	Open	Minor
3.6	Variable and function naming should be clearer.	Open	Minor
3.7	Token holders must trust the token controller owner fully.	Open	Minor

3 Issue Detail

3.1 When checking proof of use, an unconfirmed whitelisted pair is used.

Severity	Status	Link	Remediation Comment
Major	Open	issues/25	The issue is currently under review

Description

When a token holder is part of a whitelisted pair, its proof of use is the sum of both members of the pair. The following code is meant to determine the other member of a whitelisted pair, but it fails to check that the pairing has been confirmed.

contracts/FoamTokenController.sol:L120

```
        address secondAddress = whitelisted[_contributor];
```

An attacker can exploit this bug by calling `setWhitelisted` and passing the address of any non-contributor that has already completed the proof of use requirements. This would instantly allow the attacker to make unrestricted token transfers.

Remediation

Any time a whitelisted pair is considered, it's critical that only confirmed pairings are considered. This can be accomplished after the above code by checking that `whitelisted[secondAddress] == _contributor`.

3.2 Whitelisting mechanism allows immediate trading, without proof of use.

Severity	Status	Link	Remediation Comment
Major	Open	issues/24	The issue is currently under review

Description

The FOAM token controller places restrictions on how tokens can be transferred during the first year after the platform's launch. The "whitelisted pairs" mechanism, which is intended to allow token holders to use their tokens from a secondary account, has the side effect of allowing early token trades.

A token holder can sell some amount of their tokens to a buyer by whitelisting the buyer and making the transfer. The easiest way to guarantee such a trade would be to make the buyer be a contract that transfers funds on receipt of the tokens.

More sophisticated schemes would involve whitelisting a contract that acts on the seller's behalf. With a little coordination, a new token W-FOAM ("wrapped FOAM") could be created, where each W-FOAM can be traded for a FOAM token once tokens can be freely traded (e.g. one year after launch). Token holders could lock up their FOAM in contracts that would grant them W-FOAM instead, and W-FOAM could be freely traded on existing token exchanges.

Remediation

Removing the "whitelisting pairs" feature would mitigate this vulnerability, but it's important to note that no remediation will fully disallow this sort of thing. Determined token holders can always make such financial deals outside of this and associated contracts.

3.3 Blacklist feature is untested

Severity	Status	Link	Remediation Comment
Medium	Open	issues/27	The issue is currently under review

Description

Test coverage reveals that the blacklisting feature is not covered in any existing tests.

Remediation

Tests should be written that exercise the blacklist. Suggested tests: adding and removing addresses from the list, positive and negative tests for transfers from blacklisted accounts.

3.4 Whitelisting logic should be simplified.

Severity	Status	Link	Remediation Comment
Medium	Open	issues/26	The issue is currently under review

Description

The whitelisted pair mechanism contains subtle logic that makes it hard to use it correctly. Creating a whitelisted pair is a two-step process:

1. First, an account that contributed to the token sale must call `setWhitelisted(secondAccount)`.
2. Second, that other account must call `confirmWhitelisted(firstAccount)`.

The `whitelisted` mapping serves double duty: `whitelisted[a] == b` by itself does not necessarily mean that `a` and `b` form a whitelisted pair. Instead, it could be that `a` completed step 1 above, but the pairing was never confirmed.

This subtlety directly led to the major vulnerability https://github.com/ConsenSys/foam-controller-audit-2018-08-24/issues/25.

Remediation

Right now, all consumers of the `whitelisted` mapping must be careful to check both directions of a pairing to verify that it has been confirmed. This could be simplified by splitting `whitelisted`'s double duty into two different mappings as follows:

1. `proposedPair[a] == b` means that `a` has called `setWhitelisted(b)`, but `b` has not yet confirmed that pairing.
2. `paired[a] == b` means that `a` and `b` form a confirmed whitelisted pair. This is set in both directions in `confirmWhitelisted()`.

All other functions just need to look at `paired[x]` to see confirmed pairs only.

3.5 Token controller contract assumes behavior of other external contracts

Severity	Status	Link	Remediation Comment
Minor	**Open**	issues/30	The issue is currently under review

Description

The token controller contract makes external calls to the `sale` and `registry` contracts. These are trusted contracts which are assumed to behave in certain ways. If these assumptions are violated in the future, that could lead to misbehavior on the part of the token controller.

Examples

contracts/FoamTokenController.sol:L45-L46

```
        saleTokensPerUnit = sale.saleTokensPerUnit();
        extraTokensPerUnit = sale.extraTokensPerUnit();
```

These two variables from the `sale` contract are only read once, at the time of deployment. There's an implicit assumption that they're constant, which they probably are in this specific instance.

---

contracts/FoamTokenController.sol:L122

```
        uint256 contributed = sale.unitContributions(_contributor);
```

contracts/FoamTokenController.sol:L132-L133

```
        uint256 tokensStaked = registry.totalStaked(_contributor);
        uint256 PoICreated = registry.numApplications(_contributor);
```

`sale.unitContributions()`, `registry.totalStaked()`, and `registry.numApplications()` are all called without following the Checks-Effects-Interactions pattern. If they in turn were to make calls to untrusted contracts, this might introduce a reentrancy vulnerability.

Further, the "whitelisted pair" mechanism assumes that accounts are in two disjoint sets: token sale contributors and non-contributors. If `sale.unitContributions()` can change after the token controller is deployed, that assumption may no longer hold.

Remediation

No immediate action is required, but it's a good idea to document these requirements. Larger contract systems tend to accrue hidden dependencies like this can lead to surprising vulnerabilities in the future.

References

- Security Considerations — Solidity 0.4.25 documentation

3.6 Variable and function naming should be clearer.

Severity	Status	Link	Remediation Comment
Minor	**Open**	issues/29	The issue is currently under review

Description

Descriptive naming is an important way to improve code readability. This code contains several poorly named variables and functions which make it hard to understand their purpose and proper use.

Examples

contracts/FoamTokenController.sol:L24

```
    mapping(address => address) public whitelisted;
```

`whitelisted` is a poor name, especially when juxtaposed with `isBlacklisted`. Something like `pairedAccounts` would be more descriptive.

---

contracts/FoamTokenController.sol:L32

```
    address acceptedAddress = '0x36A9b165ef64767230A7Aded71B04F0911bB1283';
```

`acceptedAddress` might be better named `blacklistDestinationAddress` to make explicit its relationship with the blacklisting feature.

---

contracts/FoamTokenController.sol:L49

```
    function setWhitelisted(address _whitelisted) public {
```

`setWhitelisted` is a confusing name. Not only is the term "whitelisted" confusing, but this function is merely the first step towards establishing a whitelisted pair. Perhaps something like `proposePairing` would be clearer.

---

contracts/FoamTokenController.sol:L119

```
    function purchaseCheck(address _contributor) internal view returns(bool) {
```

`purchaseCheck` would be better named something like `hasMetProofOfUse`.

Remediation

It's recommended that a pass be taken through the entire contract to establish more descriptive names.

3.7 Token holders must trust the token controller owner fully.

Severity	Status	Link	Remediation Comment
Minor	Open	issues/28	The issue is currently under review

Description

The token controller contract gives the owner full control over who can transfer tokens and when. Functions like `blacklistAddresses` and `setPlatformLaunchDate` make such changes easy for the token controller owner.

The owner of the token contract can also swap in an entirely new token controller contract at any time, containing custom logic for when to allow or deny transfers.

This also makes the contract owner's private key a high-value target, as anyone who acquires that key gains control over the tokens.

Remediation

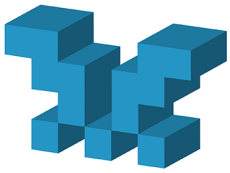
Transparency about the trust model is encouraged. Token holders should understand that the token's functionality is entirely controlled by the the accounts that own the contracts involved.

Special care should be taken to protect the private key for the owning account, including perhaps deploying from a multisig contract to mitigate the damage from key compromise.

4 Tool based analysis

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

4.1 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.

4.2 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.

4.3 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.

4.4 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.

5 Test Coverage Measurement

Testing is implemented using Truffle. 50 tests are included in the test suite and they all pass.

The Solidity-Coverage tool was used to measure the portion of the code base exercised by the test suite, and identify areas with little or no coverage. Specific sections of the code where necessary test coverage is missing are included in chapter 3 - Issues.

It's important to note that "100% test coverage" is not a silver bullet. Our review also included a inspection of the test suite, to ensure that testing included important edge cases.

The state of test coverage at the time of our review can be viewed by opening the `index.html` file from the coverage report directory in a browser.

Appendix 1 - File Hashes

Find the full list of files in the scope including SHA1 hashes in the Surya tool directory.

Appendix 2 - Severity

A.2.1 - 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.2.2 - 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.2.3 - 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.2.4 - 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 3 - 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.