Solidity Compiler Audit

OPENZEPPELIN SECURITY | NOVEMBER 1, 2018

Security Audits

The Augur team and the Ethereum Foundation (through a joint grant) asked us to review and audit the Solidity compiler. We looked at the code and now publish our results.

The audited project can be found in the ethereum/solidity GitHub repository. The version used for this report is commit `e67f0147998a9e3835ed3ce8bf6a0a0c634216c5` (tag v0.4.24).

The full report can be found here, and a list of the issues ordered by severity can be found next.

Critical Severity

- CRITICAL: Incorrect library addresses can be injected while linking.
- CRITICAL: Comments can be disguised as executable code.

High Severity

- HIGH: Model is very complex and could use more documentation.
- HIGH: Known issues only emit warnings for backwards compatibility.
- HIGH: There is no report of unit test coverage.
- HIGH: Low unit test coverage.
- HIGH: All strings are UTF-8.
- HIGH: Modifiers can be overridden with no special syntax or warnings.
- HIGH: solc-js output with optimizations is non-deterministic in some environments.
- HIGH: No error message on uninitialized storage references.
- HIGH: Missing return statement on a function does not issue an error.

- MEDIUM: Insecure system call may lead to command execution.
- MEDIUM: Swarm hash implementation is outdated.
- MEDIUM: Fallback mechanism in imports is not working properly.
- MEDIUM: Coinspect audit still has unaddressed issues.
- MEDIUM: Bus factor is 2.
- MEDIUM: There is no code of conduct.
- MEDIUM: There is no clear test structure.
- MEDIUM: There is no intermediate language.
- MEDIUM: The syntax for the fallback function is prone to confusion.
- MEDIUM: Some public functions cannot be made external.
- MEDIUM: State variables can be shadowed.
- MEDIUM: Optional optimizations may not be safe.
- MEDIUM: Optimizations code in the assembler (libevmasm) is hard to read.
- MEDIUM: Fragile code in the CSE optimizer.
- MEDIUM: All optimizations are very low level.
- MEDIUM: Modifiers can return.
- MEDIUM: No error when externally calling contract code from a constructor.
- MEDIUM: No dead code warning.
- MEDIUM: Crash when trying to declare an already declared variable with the same name.
- MEDIUM: Crash when converting signed rational using ABIEncoderV2

Low severity

- LOW: Coding style hinders readability and may lead to programming errors.
- LOW: Insecure string handling.
- LOW: The quality of sourcemaps could be improved.
- LOW: There are many assertThrow usages without a message
- LOW: Storage of small value types is unnecessarily costly.
- LOW: There are issues tagged as Soon that have not been updated in a long time.
- LOW: There are many untriaged issues.
- LOW: There are few issues tagged as Good first issue.
- LOW: There are many open pull requests with multiple comments.

- LOW: There is a lot of inconsistency on the Julia, IULIA, Yul name.
- LOW: The status of Yul is not clear.
- LOW: The main project README is missing important information.
- LOW: The main page of the user documentation has many links.
- LOW: It is unclear which files are included in a GitHub release.
- LOW: Whiskers is documented as part of the contribution guidelines.
- LOW: The process for helping with translations is not documented.
- LOW: Documentation translations are hosted on independent sites.
- LOW: There is no documentation explaining how to help testing the nightly build.
- LOW: There is no documentation on how tests are run on Continuous Integration.
- LOW: There is no clear documentation for experimental features.
- LOW: No documentation available for libsolc.
- LOW: README for Yul optimizations is incomplete.
- LOW: Missing information for successfully building Solidity's fuzzer AFL.
- LOW: soltest custom command line arguments are not listed in help.
- LOW: There is no clear documentation about the constructor not being part of the deployed code.
- LOW: Contracts from external projects are duplicated in the Solidity code repository.
- LOW: Some tests are run twice on different Continuous Integration systems.
- LOW: There are no static tests enforcing a consistent code style.
- LOW: It is very difficult to run tests locally.
- LOW: Building in some Linux distributions fails.
- LOW: Missing file on compilation when using SANITIZE.
- LOW: Insecure environment variable handling
- LOW: No errors on missing output option.
- LOW: Inconsistent AST output.
- LOW: Confusing options naming.
- LOW: Undocumented clone contract feature.
- LOW: General CLI inconsistencies and confusing options.
- LOW: No mechanism to prevent functions from being overridden.
- LOW: Invalid UTF-8 sequences are allowed in comments.
- LOW: It is not possible to declare constant variables inside functions.

- LOW: Erroneous mutability detection when dead code is involved.
- LOW: Misleading error message on overload resolution failure.
- LOW: Misleading error when externally referencing a state variable.
- LOW: Misleading error when internally calling an external function.
- LOW: Fuzzer.cpp and solfuzzer have counterintuitive naming.
- LOW: AFL example from the documentation doesn't work.
- LOW: Fuzz testing scheduling and visibility
- LOW: Crash when requested type is not present.
- LOW: Crash when accessing empty name variable slot.
- LOW: Crash when type not set for parameter return value.
- LOW: Crash when type not set for parameter function value.
- LOW: Crash when accessing a *slot of a function in assembly block*.
- LOW: Crash when calling a non callable type on a non primitive type double assignment.
- LOW: Crash when using assembly jump instruction inside a constructor or function with same name as contract.
- LOW: Crash when declaring external function with array of struct that possesses arrays.
- LOW: Crash when using struct as external function parameter using ABIEncoderV2.
- LOW: Crash when converting fixed point type using ABIEncoderV2.
- LOW: Crash when array index value is too large.
- LOW: High CPU usage on conversion between numeric literal and others.
- LOW: High CPU usage when using large variable names.

Notes

- NOTE: Non-functional requirements.
- NOTE: Micropayment Channel example is not written.
- NOTE: Consider reviewing the language design process and adding high-level goals.
- NOTE: Tests hang if cpp-ethereum is not in $PATH.
- NOTE: The help string for the—libraries option is wrong.
- NOTE: The deprecated var keyword is documented.
- NOTE: Deprecated constructors found in examples.
- NOTE: Warnings for unassigned arrays are not truncated.

recommendations on how to fix them. Some additional changes were proposed to follow best practices and reduce potential attack surface.

Update: All critical and high severity issues were fixed or addressed by the Solidity team.

Note that as of the date of publishing, the above review reflects the current understanding of known security patterns as they relate to the Solidity compiler. We have not reviewed the Augur project. The above should not be construed as investment advice or an offering of tokens. For general information about smart contract security, check out our thoughts here.

Related Posts

Beefy Zap Audit

BeefyZapRouter serves as a versatile intermediary designed to execute users' orders through routes...

Security Audits

OpenBrush Contracts Library Security Review

OpenBrush is an open-source smart contract library written in the Rust programming language and the...

Security Audits

Linea Bridge Audit

Linea is a ZK-rollup deployed on top of Ethereum. It is designed to be EVM-compatible and aims to...

Security Audits

OpenZeppelin