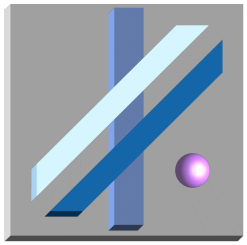

Fixed Point Solutions, LLC

solmate Smart Contract Library Assessment

2021/12/31
Prepared by: Kurt Barry

1. Scope

The contents of src/auth/, src/tokens, and src/utils were audited at the following commit hash:
https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/

Additionally, the following commits were reviewed on a stand-alone basis:
https://github.com/Rari-Capital/solmate/commit/a0253999252f9a253fefa960be6ad77d2ed1eaa7

The following files were reviewed at the linked commits on a stand-alone basis:
https://github.com/Rari-Capital/solmate/blob/0216f06c9f208ced1fdb3187d4a7a17aedec84a3/src/tokens/ERC721.sol
https://github.com/Rari-Capital/solmate/blob/83607e0a9e10105ee1f5cf5eaa0ce61026c8eb32/src/auth/authorities/MultiRolesAuthority.sol
https://github.com/Rari-Capital/solmate/blob/5d1533cbe94adf848bf322e1c9a4a43bb159bb5b/src/tokens/ERC1155.sol

Commits addressing findings in this report were reviewed as well.

2. Limitations

No assessment can guarantee the absolute safety or security of a software-based system. Further, a system can become unsafe or insecure over time as it and/or its environment evolves. This assessment aimed to discover as many issues and make as many suggestions for improvement as possible within the specified timeframe. Undiscovered issues, even serious

ones, may remain. Issues may also exist in components and dependencies not included in the assessment scope.

3. Findings

Findings and recommendations are listed in this section, grouped into broad categories. It is up to the team behind the code to ultimately decide whether the items listed here qualify as issues that need to be fixed, and whether any suggested changes are worth adopting. When a response from the team regarding a finding is available, it is provided.

Findings are given a severity rating based on their likelihood of causing harm in practice and the potential magnitude of their negative impact. Severity is only a rough guideline as to the risk an issue presents, and all issues should be carefully evaluated.

Severity Level Determination		Impact		
		High	Medium	Low
Likelihood	High	**Critical**	**High**	**Medium**
	Medium	**High**	**Medium**	**Low**
	Low	**Medium**	**Low**	**Low**

Issues that do not present any quantifiable risk (as is common for issues in the Code Quality category) are given a severity of **Informational**.

3.1 Security and Correctness

Findings that could lead to harmful outcomes or violate the intentions of the system.

SC.1 Behavior of SSTORE2 Library is Sensitive to Compiler Version Used

Severity: Low

Code Location:
https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/SSTORE2.sol#L2

Description: The compiler pragma for SSTORE2.sol allows any Solidity version 0.7.0 or higher to be used. However, the code contains mathematical operations which will have overflow checking in versions of Solidity >=0.8.0, but there will be no overflow checking when a 0.7.X version is used. This can result in silent errors and unpredictable behavior, particularly since `extcodecopy` will not raise any error if instructed to read past the end of a contract's bytecode. It is recommended to ensure that the code will behave identically no matter which compiler

version is used (or to restrict the pragma to whichever compiler versions give the intended behavior). Note further that the `<address>.code` syntax is not present prior to 0.8.0, so as written the code will not compile with 0.7.X.

Response: Pragma adjusted to `>=0.8.0` in commit 17ab906bb9a4275eb2b05440d179b423668dc003.

SC.2 A Misbehaving `authority` in `Auth` Cannot Be Replaced

Severity: Low

Code Location:
https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/auth/Auth.sol#L49

Description: The authorization judgment of `authority`, if present, always takes precedence over the `owner` of a contract inheriting from `Auth`. The `setAuthority` function in particular relies on this behavior, so a faulty or malicious `authority` can prevent its own replacement. If the `setAuthority` function instead gave precedence to whether `msg.sender == owner` in its authorization check, a misbehaving `authority` could be replaced.

Response: Fixed.

SC.3 Incorrect ERC165 Interface ID in ERC1155 Implementation

Severity: Informational

Code Location:
https://github.com/Rari-Capital/solmate/blob/5d1533cbe94adf848bf322e1c9a4a43bb159bb5b/src/tokens/ERC1155.sol#L144

Description: Per https://eips.ethereum.org/EIPS/eip-165, the correct interface id for ERC165 is `0x01ffc9a7`; the value used here (`0x5b5e139f`) corresponds to that for ERC721Metadata.

Response: Fixed in commit 3f9dcf84e68f2552ff193291cfd7bd9bc0976e07.

3.2 Usability and Incentives

Findings that could lead to suboptimal user experience, hinder integrations, or lead to undesirable behavioral outcomes.

U.1 Cannot Specify an Ether Endowment with CREATE3 Implementation

Severity: Informational

Code Location:
https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/CREATE3.sol#L26

Description: The CREATE opcode supports bestowing an initial Ether balance onto a contract, and the deployment proxy code has full support for passing this argument, but no way to utilize this functionality is provided by this library. This is a tradeoff that limits applicability of the library in exchange for a small gas savings (namely by omitting an argument to `deploy`). As a matter of principle, libraries usually bias towards generality so that they can be as widely-used as possible. It also precludes compatibility with the proposed behavior of EIP 3171, which includes an Ether endowment (https://github.com/ethereum/EIPs/pull/3171/files). This limitation at the very least should be clearly documented so potential consumers of the library are more likely to realize whether or not they can use it as-is early in their development cycle. It would also be possible to add a second `deploy` function that allows specifying an Ether value and consumers of the library can choose which to utilize based on their needs.

Response: Fixed in commit 5db685309d4ad25f4c90e87c8d5189f8e1a7955b.

U.2 Contracts Inheriting from `ERC20` Might Implement Functionality that Makes Uses of `unchecked` Unsound

Severity: Informational

Code Location:
[1]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/tokens/ERC20.sol#L82
[2]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/tokens/ERC20.sol#L104
[3]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/tokens/ERC20.sol#L174
[4]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/tokens/ERC20.sol#L186

Description: A child contract could directly modify balances or `totalSupply` in a way that makes these unchecked blocks unsound. A concrete way this could occur is via a consumer trying to use the popular trick of setting the contract's own balance to `type(uint256).max` to prevent token transfers to the contract itself via an implicit overflow check on balance increases. Users of the library should be aware of the assumptions made in `ERC20` and take not to violate them, or to override functions as needed. This efficiency versus safeguards tradeoff is in-line with solmate's general design philosophy, but ideally it should be made clear to consumers via comments or other documentation.

Response: Documented in commit 1595c321d93e6ac7d76d3a92b4f15bb50560ef6c.

U.3 Event Emitted From `Trust` Even When the Trustedness of an Address is Unchanged

Severity: Informational

Code Location:
https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/auth/Trust.sol#L20

Description: These events are emitted even when the trust status of the subject address has not changed (e.g. re-trusting and already-trusted address). This makes interpreting an emitted event more challenging e.g. for monitoring software, which must maintain or query the previous state of the contract to determine whether an event represents a meaningful change in trust surface.

Response: Obsolete (`Trust` removed).

3.3 Gas Optimizations

Findings that could reduce the gas costs of interacting with the protocol, potentially on an amortized or averaged basis.

G.1 Unnecessary Free Memory Pointer Updates

Severity: **Low**

Code Location:
[1]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/SafeTransferLib.sol#L44
[2]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/SafeTransferLib.sol#L74
[3]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/SafeTransferLib.sol#L103

Description: It is only necessary to update the free memory pointer when modifying memory in assembly and then relying on those modifications to persist after returning to Solidity execution, which is not done in any of these functions. Further, Solidity itself makes no guarantee that memory at or beyond the free memory pointer will be zeroed out (emphasis added):

“There are some operations in Solidity that need a temporary memory area larger than 64 bytes and therefore will not fit into the scratch space. They will be placed where the free memory points to, but given their short lifetime, the pointer is not updated. The memory may or may not be zeroed out. Because of this, **one should not expect the free memory to point to zeroed out memory**.”

(source: https://docs.soliditylang.org/en/v0.8.10/internals/layout_in_memory.html). Thus leaving non-zero values at and beyond the free memory pointer should not break any legitimate assumptions made by calling code.

Response: Fixed in commit 134f10afd9c996cee217aec6274839f56c742eac.

G.2 Excess Space Reserved in Memory When Reading Bytecode in `SSTORE2`

Severity: Low

Code Location:
[1]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/SSTORE2.sol#L56

Description: In the calculation of the new location of the free memory pointer, it is unnecessary to add `start` as the space reserved in memory only needs to be large enough for a single word plus the number of bytes to be read plus padding for word-alignment. There is no correctness issue here as reserving excess space is always sufficient for the data to be read; however, elimination of this numeric operation should save a small amount of gas.

Response: Fixed in commit 5546cd3cf78f43366068ce6cb1b7f58c8d9fca65.

G.3 `fpow` Assembly Optimizations

Severity: Low

Code Location:
[1]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/FixedPointMathLib.sol#L87
[2]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/FixedPointMathLib.sol#L89
[3]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/FixedPointMathLib.sol#L91
[4]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/FixedPointMathLib.sol#L94

Description:
[1] can replace `div(baseUnit, 2)` with the cheaper `shr(1, baseUnit)`
[2] can replace `div(n, 2)` with the cheaper `shr(1, n)`
[3] same as [2]
[4] can replace `iszero(eq(div(xx, x), x))` with the cheaper `shr(128, x)`

Response: Fixed in commit 0265de1f3d7b65a9961b0499e69d9471c199d1ac.

3.4 Code Quality

CQ.1 Inconsistent Use of Numeric Base to Express Pointer Offsets in SSTORE2

Severity: **Informational**

Code Location:
[1]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/SSTORE2.sol#L20
[2]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/SSTORE2.sol#L56
[3]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/SSTORE2.sol#L58

Description: At [1] “32” (decimal) is used, whereas for [2] and [3], “0x20” (hexadecimal) is used. Consistency in numeric base would make the code more readable and maintainable.

Response: Fixed in commit 5546cd3cf78f43366068ce6cb1b7f58c8d9fca65.

CQ.2 Comment Could Be Added to Clarify Prepending a Zero-Byte in `SSTORE2`

Severity: **Informational**

Code Location:
https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/utils/SSTORE2.sol#L10

Description: The zero byte is prepended here to prevent unintentional execution of the stored data, as is noted in a comment in the code this library was inspired by:

https://github.com/0xsequence/sstore2/blob/master/contracts/SSTORE2.sol#L22.

(It works because 0 represents the STOP opcode of the EVM.) Including the comment here as well would make the code easier to understand for anyone using, modifying, or maintaining it.

Response: Fixed in commit fc0d77b124c200993c441d3f7612e8731837fd1c.

CQ.3 Excessive Casting In `RolesAuthority`

Severity: **Informational**

Code Location:
[1]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/auth/authorities/RolesAuthority.sol#L36
[2]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/auth/authorities/RolesAuthority.sol#L56
[3]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/auth/authorities/RolesAuthority.sol#L99
[4]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/auth/authorities/RolesAuthority.sol#L119

Description: The expression `uint256(uint256(2)**uint256(role))` in these four code locations can be replaced with just `2**uint256(role)`, improving readability.

Response: Fixed in commit 7c5cafff2f9e04a46c71719710aee68d34eee8c5.

CQ.4 Inconsistent Mapping Naming In `RolesAuthority`

Severity: **Informational**

Code Location:
[1]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/auth/authorities/RolesAuthority.sol#L32
[2]https://github.com/Rari-Capital/solmate/blob/938b2d6925e24de7542bce4b08f3f03988a82245/src/auth/authorities/RolesAuthority.sol#L46

Description: The mappings `getUserRoles` and `getRoleCapabilities` use opposite naming conventions. The former maps each user to a set of roles, and the key type precedes the value type in the name. The latter maps each capability (an ordered pair of target contract address and function signature) to a set of roles, but the value type precedes the key type. The code would be easier to read and maintain if a consistent naming convention were used.

Response: Addressed in commit a3d179efcb3a7d954cf0b51a7b6bf0ee90c84dea.

4. Notes

This section contains general considerations for interacting with or maintaining the system and various conclusions reached or discoveries made during the course of the assessment. Whereas findings generally represent things for the team to consider changing, notes are more informational and may be helpful to those who intend to interact with the system.

4.1 Considerations Regarding Fork Protection in `ERC20.permit`

The solmate `ERC20` contract includes EIP-2612 functionality which allows token holders to allow others to operate on their balance without spending gas themselves. This involves creating a signed blob of data, the "permit", that must be verified by the contract. The audited commit includes protection against replaying commits from one chain against a fork of it (or any other chain where sufficient conditions are satisfied such that a particular permit is valid). This is implemented by recalculating the `DOMAIN_SEPARATOR` if the chainid differs from that observed at the time of contract construction. This slightly increases gas costs, so the risks of removing this protection (by fixing the `DOMAIN_SEPARATOR` value in the constructor) were discussed by the auditor and the client. The following considerations were produced:

1. The risk of an Ethereum mainnet hardfork is exceedingly low due to bridged or centralized assets like WBTC, USDT, and USDC that would be forced to choose which ledger they would honor, destroying most or all value on the non-chosen fork. This creates a strong disincentive for forking in the first place.

2. The first consideration extends to any chain that also has large quantities of bridged or centralized assets.

3. There may be some risk to technical users if permits used to test against local forks of a mainnet chain are leaked (for example by being checked in to GitHub as part of a test suite) or stolen (e.g. by malware with the ability to capture data from files or terminal output).

4. Some commonly-used tokens (e.g. DAI and all Uniswap V2 LP tokens) already lack true fork protection (i.e. they fix the `DOMAIN_SEPARATOR` value at the time of construction), yet the exploit from the third consideration has not been observed in the wild for the these tokens (at least, neither to the knowledge of the auditor nor the client). While this does not mean it will not be observed in the future, or become more common if already happening at a low rate, it does suggest the current risk of the third consideration is quite low.

4.2 ERC20 `approve` Race Condition

When one address sends a transaction to change the `transferFrom` allowance granted to another address, the other address can frontrun this change and claim both approval amounts. This is a well-known issue with the ERC20 standard (more information: https://swcregistry.io/docs/SWC-114). Most integrators know how to deal with this issue already; thus, it is not considered a "finding" in the formal sense and no specific mitigation is recommended. One possible enhancement would be to add `increaseAllowance`/`decreaseAllowance` functions or a `safeApprove` method that checks that the current allowance matches an expectation prior to applying the update.

4.3 ERC721 Standard Divergence

The ERC721 standard specifies that `balanceOf` should throw when queried on the zero address; the `solmate` implementation does not. This is a gas optimization and unlikely to cause issues in practice, but library consumers should keep it in mind and warn integrators as necessary.

A warning was added by the Rari team in commit [824bd883069651d1751b6606d093e783cb6e4905].