



Security Assessment & Formal Verification Report v3



August 2024

Prepared for
Safe Ecosystem Foundation

Table of content

Project Summary	4
Project Scope	4
Project Overview	4
Findings Summary	5
Severity Matrix	5
Detailed Findings	6
Low Severity Issues	7
L-01. The Solidity version used (0.7.6) contains known severe issues	7
Informational Severity Issues	8
I-01. Some compiler versions will throw the "Invalid implicit conversion from address to address payable requested" error	8
I-02. Usage of floating pragma is not recommended on top-level contracts	10
I-03. Typos	11
I-04. Clarify some comments	12
I-05. Unnecessary casting	13
I-06. Internal and private variables and functions names should begin with an underscore	14
Gas Optimization Recommendations	15
G-01. Caching storage variables to save gas	15
G-02. Emit the existing memory variable instead of reading from storage	15
G-03. Unnecessary external calls	16
G-04. Functions can be marked payable to save 24 gas	16
Formal Verification	17
Verification Notations	17
Formal Verification Properties	18
SafeMigration.sol	18
P-01. Immutability of MIGRATION_SINGLETON address	18
P-02. All non-view functions should revert if called directly	19
P-03. All migration functions update correctly the Safe's singleton address	20
P-04. Relevant migration functions update correctly the Safe's fallbackHandler address	21
SafeToL2Setup.sol	22
P-01. Immutability of _SELF address	22
P-02. All non-view functions should revert if called directly	23
P-03. The setup function setupToL2() updates correctly the Safe's singleton address	24
P-04. delegateCall to setupToL2() can succeed only if Safe's nonce is zero	25
SafeToL2Migration.sol	26
P-01. Immutability of MIGRATION_SINGLETON address	26
P-02. All non-view functions should revert if called directly	27
P-03. The migration function migrateFromV111() updates correctly the Safe's singleton and fallbackHandler addresses	28

P-04. The migration function migrateToL2() updates correctly the Safe's singleton address.....	29
P-05. delegateCall to migrateToL2() or migrateFromV111() can succeed only if Safe's nonce is correct.....	30
Disclaimer.....	31
About Certora.....	31

Project Summary

Project Scope

Project Name	Repository (link)	Audited Commits	Platform
SafeMigration	https://github.com/safe-global/safe-smart-account/blob/main/contracts/libraries/SafeMigration.sol	07d4fc7 – initial B541cd7 – latest including fixes	EVM/Solidity 0.7
SafeToL2Setup	https://github.com/safe-global/safe-smart-account/blob/main/contracts/libraries/SafeToL2Setup.sol	07d4fc7 – initial B541cd7 – latest including fixes	EVM/Solidity 0.7
SafeToL2Migration	https://github.com/safe-global/safe-smart-account/blob/main/contracts/libraries/SafeToL2Migration.sol	07d4fc7 – initial B541cd7 – latest including fixes	EVM/Solidity 0.7

Project Overview

This document describes the specification and verification of **Safe's Migration and Setup Contract** using the Certora Prover and manual code review findings. The work was undertaken from **Aug 18, 2024** to **Aug 23, 2024**.

The following contract list is included in our scope:

contracts/libraries/SafeMigration.sol
contracts/libraries/SafeToL2Setup.sol
contracts/libraries/SafeToL2Migration.sol

The Certora Prover demonstrated that the implementation of the **Solidity** contracts above is correct with respect to the formal rules written by the Certora team. In addition, the team performed a manual audit of all the Solidity contracts. During the verification process and the manual audit, the Certora team discovered bugs in the Solidity contracts code, as listed on the following page.

Findings Summary

The table below summarizes the findings of the review, including type and severity details.

Severity	Discovered	Confirmed	Fixed
Critical	0	0	0
High	0	0	0
Medium	0	0	0
Low	1	1	0
Informational	6	6	4
Total	7	7	4

Severity Matrix

Impact	High	Medium	High	Critical
	Medium	Low	Medium	High
	Low	Low	Low	Medium
		Low	Medium	High
Likelihood				

Detailed Findings

ID	Title	Severity	Status
L-01	The Solidity version used (0.7.6) contains known severe issues	Low	Acknowledged
I-01	Some compiler versions will throw the "Invalid implicit conversion from address to address payable requested" error	Informational	Fixed
I-02	Usage of floating pragma is not recommended on top level contracts	Informational	Acknowledged
I-03	Typos	Informational	Fixed
I-04	Clarify some comments	Informational	Fixed
I-05	Unnecessary casting	Informational	Fixed
I-06	Internal and private variables and functions names should begin with an underscore	Informational	Acknowledged

Low Severity Issues

L-01. The Solidity version used (0.7.6) contains known severe issues

Description: The `hardhat.config.ts` file declares that 0.7.6 would be the default Solidity version:

File: `hardhat.config.ts`

```
44: const primarySolidityVersion = SOLIDITY_VERSION || "0.7.6";
```

However, several known issues exist in this version:

https://github.com/ethereum/solidity/blob/develop/docs/bugs_by_version.json#L1702-L1714

- FullInlinerNonExpressionSplitArgumentEvaluationOrder
- MissingSideEffectsOnSelectorAccess
- AbiReencodingHeadOverflowWithStaticArrayCleanup
- DirtyByteArrayToStorage
- DataLocationChangeInInternalOverride
- NestedCalldataArrayAbiReencodingSizeValidation
- SignedImmutables
- ABIDecodeTwoDimensionalArrayMemory
- KeccakCaching

Their details can be read here: <https://docs.soliditylang.org/en/latest/bugs.htm> |

Customer's response: This is the standard version we use across `safe-smart-account`, but based on the known vulnerabilities, we don't see any concerns in terms of the migration contracts.

Informational Severity Issues

I-01. Some compiler versions will throw the "Invalid implicit conversion from address to address payable requested" error

Description: The following event has the line `address payable refundReceiver`:

```
File: SafeToL2Migration.sol
45:     event SafeMultiSigTransaction(
46:         address to,
47:         uint256 value,
48:         bytes data,
49:         Enum.Operation operation,
50:         uint256 safeTxGas,
51:         uint256 baseGas,
52:         uint256 gasPrice,
53:         address gasToken,
54:         address payable refundReceiver,
55:         bytes signatures,
56:         // We combine nonce, sender and threshold into one to avoid stack too deep
57:         // Dev note: additionalInfo should not contain `bytes`, as this complicates decoding
58:         bytes additionalInfo
59:     );
```

However, the incorrect type is used in the `emit` statement

```
File: SafeToL2Migration.sol
083:     function migrate(address l2Singleton, bytes memory functionData) private {
...
089:         // Simulate a L2 transaction so Safe Tx Service indexer picks up the Safe
090:         emit SafeMultiSigTransaction(
091:             MIGRATION_SINGLETON,
092:             0,
093:             functionData,
094:             Enum.Operation.DelegateCall,
095:             0,
096:             0,
097:             0,
098:             address(0),
- 099:             address(0),
+ 099:             payable(address(0)),
100:             "", // We cannot detect signatures
101:             additionalInfo
102:         );
```


Customer's response: Fixed in commit [b541cd7](#) and already present in the `main` branch.

I-02. Usage of floating pragma is not recommended on top-level contracts

Description: Non-library/interface files should use fixed compiler versions, not floating ones:

File: SafeMigration.sol

```
2: pragma solidity >=0.7.0 <0.9.0;
```

File: SafeToL2Migration.sol

```
3: pragma solidity >=0.7.0 <0.9.0;
```

File: SafeToL2Setup.sol

```
2: pragma solidity >=0.7.0 <0.9.0;
```

Customer's response: Acknowledged.

I-03. Typos

File: SafeMigration.sol

```
- 31:      * @notice Addresss of the Fallback Handler
+ 31:      * @notice Address of the Fallback Handler
```

File: SafeToL2Migration.sol

```
- 134:      * A valid and compatible fallbackHandler needs to be provided, only existence will be
checked.
+ 134:      * A valid and compatible fallbackHandler needs to be provided, only existence will be
checked.
```

Customer's response: Fixed in commit [b541cd7](#).

I-04. Clarify some comments

File: SafeToL2Migration.sol

```
- 124:          // 0xef2624ae - keccak("migrateToL2(address)")
+ 124:          // 0xef2624ae - bytes4(keccak256("migrateToL2(address)"))
...
- 154:          // 0xd9a20812 - keccak("migrateFromV111(address,address)")
+ 154:          // 0xd9a20812 - bytes4(keccak256("migrateFromV111(address,address)"))
```

Customer's response: Fixed in commit [b541cd7](#).

I-05. Unnecessary casting

Description: `singleton` is a storage variable of `address` type:

```
File: SafeStorage.sol
09: contract SafeStorage {
10:     // From /common/Singleton.sol
11:     address internal singleton;
```

Therefore it's unnecessary to cast it to the `address` type here:

```
File: SafeToL2Migration.sol
112:     function migrateToL2(address l2Singleton) public onlyDelegateCall onlyNonceZero {
- 113:         require(address(singleton) != l2Singleton, "Safe is already using the singleton");
+ 113:         require(singleton != l2Singleton, "Safe is already using the singleton");
```

Customer's response: Fixed in commit [b541cd7](#).

I-06. Internal and private variables and functions names should begin with an underscore

Description: According to the Solidity Style Guide, Non-external variable and function names should begin with an [underscore](#).

Affected code:

```
# File: contracts/libraries/SafeMigration.sol
```

```
SafeMigration.sol:109:    function hasCode(address account) internal view returns (bool) {
```

```
# File: contracts/libraries/SafeToL2Migration.sol
```

```
SafeToL2Migration.sol:83:    function migrate(address l2Singleton, bytes memory functionData)
```

```
private {
```

```
SafeToL2Migration.sol:167:    function isContract(address account) internal view returns (bool) {
```

Customer's response: Acknowledged. As we don't use that particular style format in the repo, we won't make the change.

Gas Optimization Recommendations

G-01. Caching storage variables to save gas

Description: Reading from storage is expensive and 1 SLOAD can be saved here:

File: SafeToL2Migration.sol

```
112:     function migrateToL2(address l2Singleton) public onlyDelegateCall onlyNonceZero {  
- 113:         require(address(singleton) != l2Singleton, "Safe is already using the singleton");  
+ 113:         address _singleton = singleton; // Caching singleton  
+ 113:         require(address(_singleton) != l2Singleton, "Safe is already using the singleton");  
- 114:         bytes32 oldSingletonVersion = keccak256(abi.encodePacked(ISafe(singleton).VERSION()));  
+ 114:         bytes32 oldSingletonVersion = keccak256(abi.encodePacked(ISafe(_singleton).VERSION()));
```

G-02. Emit the existing memory variable instead of reading from storage

Description: Here, at line 84, the state variable `singleton` is set to the input parameter `l2Singleton`. Those variables are never changed afterwards and are equal. Consider using `l2Singleton` to save 1 SLOAD worth of gas:

File: SafeToL2Migration.sol

```
083:     function migrate(address l2Singleton, bytes memory functionData) private {  
084:         singleton = l2Singleton;  
...  
- 103:         emit ChangedMasterCopy(singleton);  
+ 103:         emit ChangedMasterCopy(l2Singleton);  
104:     }
```

G-03. Unnecessary external calls

Description: The following line is using external calls to fetch `owners` and `threshold` but internal calls are enough (and less expensive) thanks to the `delegateCall` making the 2 `STATICCALL`-s unnecessary:

```
File: SafeToL2Migration.sol
136:     function migrateFromV111(address l2Singleton, address fallbackHandler) public onlyDelegateCall
onlyNonceZero {
137:         require(isContract(fallbackHandler), "fallbackHandler is not a contract");
138:
139:         bytes32 oldSingletonVersion = keccak256(abi.encodePacked(ISafe(singleton).VERSION()));
140:         require(oldSingletonVersion == keccak256(abi.encodePacked("1.1.1")), "Provided singleton version
is not supported");
141:
142:         bytes32 newSingletonVersion = keccak256(abi.encodePacked(ISafe(l2Singleton).VERSION()));
143:         require(
144:             newSingletonVersion == keccak256(abi.encodePacked("1.3.0")) || newSingletonVersion ==
keccak256(abi.encodePacked("1.4.1")),
145:             "Provided singleton version is not supported"
146:         );
147:
148:         ISafe safe = ISafe(address(this));
149:         safe.setFallbackHandler(fallbackHandler);
150:
151:         // Safes < 1.3.0 did not emit SafeSetup, so Safe Tx Service backend needs the event to index the
Safe
- 152:             emit SafeSetup(MIGRATION_SINGLETON, safe.getOwners(), safe.getThreshold(), address(0),
fallbackHandler);
+ 152:             emit SafeSetup(MIGRATION_SINGLETON, owners, threshold, address(0), fallbackHandler);
```

G-04. Functions can be marked payable to save 24 gas

Description: Given the nature of the migration functions: making them `payable` would save 24 gas. This is because non-payable functions get more opcodes to check that ETH was not sent.

Formal Verification

Verification Notations

Formally Verified	The rule is verified for every state of the contract(s), under the assumptions of the scope/requirements in the rule.
Formally Verified After Fix	The rule was violated due to an issue in the code and was successfully verified after fixing the issue
Violated	A counter-example exists that violates one of the assertions of the rule.

Formal Verification Properties

SafeMigration.sol

Module General Assumptions

- Loop iterations: Any loop was unrolled at most 3 times (iterations)

Contract Properties

P-01. Immutability of MIGRATION_SINGLETON address.

Status: Verified

Rule Name	Status	Description	Link to rule report
MIGRATION_SINGLETONisAlwaysCurrentContract	Verified	<i>This invariant verifies that the MIGRATION_SINGLETON address can't be overridden or replaced.</i>	Report

P-02. All non-view functions should revert if called directly.

Status: Verified

Assumptions required to pass the rule:
The invariant P-01 is required

Rule Name	Status	Description	Link to rule report
allNonViewFunctionRevert	Verified	<i>All the non-view functions will revert when called directly since those functions should only be delegateCall-ed</i>	Report

P-03. All migration functions update correctly the Safe's singleton address.

Status: Verified

Assumptions required to pass the rule:
 We are using a mock contract to represent a Safe contract by inheriting from the SafeStorage contract
 We are implementing a simplified function that delegateCalls the migration contract to perform the singleton update

Rule Name	Status	Description	Link to rule report
singletonMigrationIntegrityParametric	Verified	<p><i>All the four migration functions update correctly the Safe's singleton address to the relevant value that is expected from each function:</i></p> <ul style="list-style-type: none"> <code>migrateSingleton()</code> -> <code>SAFE_SINGLETON</code> <code>migrateWithFallbackHandler()</code> -> <code>SAFE_SINGLETON</code> <code>migrateL2Singleton()</code> -> <code>SAFE_L2_SINGLETON</code> <code>migrateL2WithFallbackHandler()</code> -> <code>SAFE_L2_SINGLETON</code> 	Report

P-04. Relevant migration functions update correctly the Safe's fallbackHandler address.

Status: Verified

Assumptions required to pass the rule:
 We are using a mock contract to represent a Safe contract by inheriting from the SafeStorage contract
 We are implementing a simplified function that delegateCalls the migration contract to perform the singleton update
 We assume the Safe's setFallbackHandler() behaves as expected by using a simplified mock version of it

Rule Name	Status	Description	Link to rule report
fallbackHandler MigrationIntegrityParametric	Verified	<p>All the four migration functions update correctly the Safe's fallbackHandler address to the relevant value that is expected from each function:</p> <ul style="list-style-type: none"> • <code>migrateSingleton()</code> -> no change • <code>migrateWithFallbackHandler()</code> -> <code>SAFE_FALLBACK_HANDLER</code> • <code>migrateL2Singleton()</code> -> no change • <code>migrateL2WithFallbackHandler()</code> -> <code>SAFE_FALLBACK_HANDLER</code> 	Report

SafeToL2Setup.sol

Module General Assumptions

- Loop iterations: Any loop was unrolled at most 3 times (iterations).

Contract Properties

P-01. Immutability of _SELF address.

Status: Verified

Rule Name	Status	Description	Link to rule report
_SELFisAlways CurrentContract	Verified	<i>This invariant verifies that the _SELF address can't be overridden or replaced and it is always the address of the verified SafeToL2Setup contract</i>	Report

P-02. All non-view functions should revert if called directly.

Status: Verified

Assumptions required to pass the rule:
The invariant P-01 is required

Rule Name	Status	Description	Link to rule report
allNonViewFunctionRevert	Verified	<i>All the non-view functions will revert when called directly since those functions should only be delegateCall-ed. In this case, there is only the setupToL2() function that is public and non view, and it must be delegateCall-ed.</i>	Report

P-03. The setup function `setupToL2()` updates correctly the Safe's singleton address.

Status: Verified

Assumptions required to pass the rule:

- We are using a mock contract to represent a Safe contract by inheriting from the `SafeStorage` contract
- We are implementing a simplified function that `delegateCalls` the setup contract to perform the singleton update

Rule Name	Status	Description	Link to rule report
theSingletonContractIsUpdatedCorrectly	Verified	<i>The <code>setupToL2()</code> function updates correctly the Safe's singleton address to the relevant value that is passed to the function. Also, for the update to succeed the <code>chainId</code> must be correct.</i>	Report

P-04. delegateCall to setupToL2() can succeed only if Safe's nonce is zero

Status: Verified

Assumptions required to pass the rule:

- We are using a mock contract to represent a Safe contract by inheriting from the SafeStorage contract
- We are implementing a simplified function that delegateCalls the setup contract to perform the singleton update

Rule Name	Status	Description	Link to rule report
nonceMustBeZero	Verified	<i>If the nonce is not zero, the call to the setupToL2() function will always revert.</i>	Report

SafeToL2Migration.sol

Module General Assumptions

- Loop iterations: Any loop was unrolled at most 3 times (iterations).

Contract Properties

P-01. Immutability of MIGRATION_SINGLETON address.

Status: Verified

Rule Name	Status	Description	Link to rule report
MIGRATION_SINGLETONisAlwaysCurrentContract	Verified	<i>This invariant verifies that the MIGRATION_SINGLETON address can't be overridden or replaced and it is always the address of the verified SafeToL2Migration contract</i>	Report

P-02. All non-view functions should revert if called directly.

Status: Verified

Assumptions required to pass the rule:
The invariant P-01 is required

Rule Name	Status	Description	Link to rule report
allNonViewFunctionRevert	Verified	<i>All the non-view functions will revert when called directly since those functions should only be delegateCall-ed.</i>	Report

P-03. The migration function `migrateFromV111()` updates correctly the Safe's singleton and fallbackHandler addresses

Status: Verified

Assumptions required to pass the rule:
 We are using a mock contract to represent a Safe contract by inheriting from the SafeStorage contract
 We are implementing a simplified function that delegateCalls the migration contract to perform the singleton update
 We assume the Safe's `setFallbackHandler()` behaves as expected by using a simplified mock version of it

Rule Name	Status	Description	Link to rule report
singletonMigrateFromV111Integrity	Verified	<i>The <code>migrateFromV111()</code> function updates correctly the Safe's singleton and fallbackHandler addresses to the relevant values that are passed to the function.</i>	Report

P-04. The migration function `migrateToL2()` updates correctly the Safe's singleton address

Status: Verified

Assumptions required to pass the rule:

- We are using a mock contract to represent a Safe contract by inheriting from the `SafeStorage` contract
- We are implementing a simplified function that `delegateCalls` the migration contract to perform the singleton update

Rule Name	Status	Description	Link to rule report
singletonMigrateToL2Integrity	Verified	<i>The <code>migrateToL2()</code> function updates correctly the Safe's singleton address to the relevant value that is passed to the function.</i>	Report

P-05. delegateCall to migrateToL2() or migrateFromV111() can succeed only if Safe's nonce is correct

Status: Verified

Assumptions required to pass the rule:
We are using a mock contract to represent a Safe contract by inheriting from the SafeStorage contract
We are implementing a simplified function that delegateCalls the migration contract to perform the singleton update

Rule Name	Status	Description	Link to rule report
nonceMustBeCorrect	Verified	<i>If the nonce is not one, any of the calls to both migration functions {migrateToL2() and migrateFromV111()} will always revert.</i>	Report

Disclaimer

The Certora Prover takes a contract and a specification as input and formally proves that the contract satisfies the specification in all scenarios. Notably, the guarantees of the Certora Prover are scoped to the provided specification and the Certora Prover does not check any cases not covered by the specification.

Even though we hope this information is helpful, we provide no warranty of any kind, explicit or implied. The contents of this report should not be construed as a complete guarantee that the contract is secure in all dimensions. In no event shall Certora or any of its employees be liable for any claim, damages, or other liability, whether in an action of contract, tort, or otherwise, arising from, out of, or in connection with the results reported here.

About Certora

Certora is a Web3 security company that provides industry-leading formal verification tools and smart contract audits. Certora's flagship security product, Certora Prover, is a unique SaaS product that automatically locates even the most rare & hard-to-find bugs on your smart contracts or mathematically proves their absence. The Certora Prover plugs into your standard deployment pipeline. It is helpful for smart contract developers and security researchers during auditing and bug bounties.

Certora also provides services such as auditing, formal verification projects, and incident response.