Omni Solidity Security Review

Auditors

Saw-mon and natalie, Lead Security Researcher

Optimum, Lead Security Researcher

Shotes, Security Researcher

Carrotsmuggler, Associate Security Researcher

Report prepared by: Lucas Goiriz

October 8, 2024

Contents

1 About Spearbit

Spearbit is a decentralized network of expert security engineers offering reviews and other security related services to Web3 projects with the goal of creating a stronger ecosystem. Our network has experience on every part of the blockchain technology stack, including but not limited to protocol design, smart contracts and the Solidity compiler. Spearbit brings in untapped security talent by enabling expert freelance auditors seeking flexibility to work on interesting projects together.

Learn more about us at spearbit.com

2 Introduction

Omni is the platform for building chain abstracted applications. By linking into each rollup, developers can source liquidity and users from the entire Ethereum ecosystem.

Disclaimer: This security review does not guarantee against a hack. It is a snapshot in time of Omni Solidity according to the specific commit. Any modifications to the code will require a new security review.

3 Risk classification

Severity level	**Impact: High**	**Impact: Medium**	**Impact: Low**
Likelihood: high	Critical	High	Medium
Likelihood: medium	High	Medium	Low
Likelihood: low	Medium	Low	Low

3.1 Impact

- High - leads to a loss of a significant portion (>10%) of assets in the protocol, or significant harm to a majority of users.
- Medium - global losses <10% or losses to only a subset of users, but still unacceptable.
- Low - losses will be annoying but bearable--applies to things like griefing attacks that can be easily repaired or even gas inefficiencies.

3.2 Likelihood

- High - almost certain to happen, easy to perform, or not easy but highly incentivized
- Medium - only conditionally possible or incentivized, but still relatively likely
- Low - requires stars to align, or little-to-no incentive

3.3 Action required for severity levels

- Critical - Must fix as soon as possible (if already deployed)
- High - Must fix (before deployment if not already deployed)
- Medium - Should fix
- Low - Could fix

4 Executive Summary

Over the course of 7 days in total, Omni engaged with Spearbit to review the omni-solidity protocol. In this period of time a total of **22** issues were found.

Summary

Project Name	Omni
Repository	omni-solidity
Commit	b41cf274
Type of Project	Bridge, Smart Contracts
Audit Timeline	Sep 3rd to Sep 10th

Issues Found

Severity	Count	Fixed	Acknowledged
Critical Risk	0	0	0
High Risk	0	0	0
Medium Risk	0	0	0
Low Risk	4	3	1
Gas Optimizations	5	4	1
Informational	13	8	5
Total	**22**	**15**	**7**

5 Findings

5.1 Low Risk

5.1.1 `execGasPrice` and `dataGasPrice` might be `0` due to division errors

Severity: Low Risk

Context: FeeOracleV1.sol#L68-L72

Description: `execGasPrice` and `dataGasPrice` might be `0` due to division errors.

Recommendation: Instead of checking whether `X.{gasPrice, toNativeRate}` are non-zero it might be best to check wether the `execGasPrice` and `dataGasPrice` are non-zero:

```
function feeFor(uint64 destChainId, bytes calldata data, uint64 gasLimit) external view returns
↪  (uint256) {
    IFeeOracleV1.ChainFeeParams storage execP = _feeParams[destChainId];
    IFeeOracleV1.ChainFeeParams storage dataP = _feeParams[execP.postsTo];

    uint256 execGasPrice = execP.gasPrice * execP.toNativeRate / CONVERSION_RATE_DENOM;
    uint256 dataGasPrice = dataP.gasPrice * dataP.toNativeRate / CONVERSION_RATE_DENOM;

    require(execGasPrice > 0, "FeeOracleV1: no fee params");
    require(dataGasPrice > 0, "FeeOracleV1: no fee params");
    // ...
```

Omni: Fixed in commit ae390993 by implementing the auditor's recommendation.

Spearbit: Verified.

5.1.2 `PortalRegistry` contract has payable function without any withdraw mechanism

Severity: Low Risk

Context: PortalRegistry.sol#L93

Description: PortalRegistry.sol#L93 contains function `bulkRegister` that is payable. However, the contract has no mechanism to move those funds anywhere and does not provide info about the utility of maintaining a balance (for gas fees, proof of value, etc..). If this function is called with `value`, that `value` will be stuck on the contract forever.

This function is only callable by the `owner`, so the risk is low.

Recommendation: Remove the `payable` keyword from the `bulkRegister` function, or document the reasoning behind maintaining an unrecoverable balance on this contract.

Omni: Fixed in commit ae390993 by removing the `payable` keyword from `bulkRegister`.

Spearbit: Fix verified.

5.1.3 Current system lacks the ability to retry failed cross chain messages

Severity: Low Risk

Context: *(No context files were provided by the reviewer)*

Description: The `OmniPortal` contract facilitates cross-chain messaging for various supported networks, enabling a wide range of applications. While some of these applications can tolerate non-atomic behavior (e.g., where the transaction on the receiving chain can fail without affecting the originating chain's success), others—such as token bridging—require atomicity to prevent potential loss of user funds.

Failures in the receiving chain can occur due to out-of-gas exceptions or subtle code errors that may not surface until after deployment. For example, `OmniBridgeNative` handles such failures gracefully by using a claimable mapping to store uncredited amounts in the event of a failure within the withdraw function. This allows users to claim funds later. However, third-party applications built on `OmniPortal` may not always implement similar fail-safes to ensure cross-chain atomicity.

Recommendation: To mitigate the risk of failed cross-chain messages, consider implementing a retry mechanism, note that it will require changing the `_exec` to allow the transaction to revert for the inner call and in addition to allow non sequential "offset" (nonce) for cross chain messages. If this feature is planned for future releases, ensure that both third-party developers building on `OmniPortal` and end users are made aware of this edge case. It's also worth noting that similar solutions, such as `LayerZero` and `Optimism`, offer retry capabilities to address these issues.

Omni: Acknowledged. Currently, retries can be implemented at the smart contract level. We are considering adding a protocol-native retry mechanism in a future release, but this is out of scope for now.

Spearbit: Acknowledged.

5.1.4 Upcast and only use multiplication to avoid overflows and imprecision

Severity: Low Risk

Context: Quorum.sol#L62

Description: The current formula used could overflow during the multiplication and also due to division the `_isQuorum` check is not quite precise since it is rounded down.

Recommendation: Instead use:

```
return votedPower * uint256(denominator) > totalPower * uint256(numerator);
```

- `forge s --diff`:

```
test_xsubmit_duplicateValidator_reverts() (gas: -57 (-0.000%))
test_xsubmit_noQuorum_reverts() (gas: -114 (-0.000%))
test_xsubmit_guzzle50_succeeds() (gas: -171 (-0.000%))
test_xsubmit_guzzle25_succeeds() (gas: -171 (-0.000%))
test_xsubmit_guzzle10_succeeds() (gas: -171 (-0.000%))
test_xsubmit_guzzle5_succeeds() (gas: -171 (-0.000%))
test_pauseAll() (gas: -171 (-0.000%))
test_xsubmit_xblock1_succeeds() (gas: -171 (-0.000%))
test_xsubmit_xblock1_chainB_succeeds() (gas: -171 (-0.000%))
test_xsubmit_guzzle1_succeeds() (gas: -171 (-0.000%))
test_xsubmit_addValidator_succeeds() (gas: -171 (-0.000%))
test_xsubmit_reentrancy_reverts() (gas: -171 (-0.000%))
test_xsubmit_invalidAttestationRoot_reverts() (gas: -171 (-0.000%))
test_xsubmit_wrongStreamOffset_reverts() (gas: -171 (-0.000%))
test_xsubmit_wrongDestChainId_reverts() (gas: -171 (-0.000%))
test_xsubmit_invalidMsgs_reverts() (gas: -171 (-0.000%))
test_xsubmit_addValidatorSet_succeeds() (gas: -342 (-0.001%))
test_xsubmit_notNewValSet_succeeds() (gas: -342 (-0.001%))
test_xsubmit_xblock2_succeeds() (gas: -342 (-0.001%))
test_xsubmit_xblock2_chainB_succeeds() (gas: -342 (-0.001%))
test_verify_powerBasedQourum_succeeds() (gas: -114 (-0.015%))
test_verify_sigsNotSorted_reverts() (gas: -57 (-0.020%))
test_verify_duplicateValidator_reverts() (gas: -57 (-0.020%))
test_verify_largeValset_succeeds() (gas: -3819 (-0.035%))
test_verify_noQuorum_fails() (gas: -114 (-0.040%))
test_verify_succeeds() (gas: -171 (-0.058%))
test_pause_unpause() (gas: -28475 (-0.293%))
test_upgrade() (gas: -42708 (-0.313%))
Overall gas change: -79448 (-0.001%)
```

Omni: Fixed in commit ae390993 by implementing the auditor's recommendation.

Spearbit: Fix verified.

5.2 Gas Optimization

5.2.1 `_feeParams[key].chainId` is redundant

Severity: Gas Optimization

Context: FeeOracleV1.sol#L32

Description: It seems redundant that the `key` for the mapping `_feeParams` is always the same as `_feeParams[key].chainId`.

Recommendation: Perhaps the field `_feeParams[key].chainId` can be removed.

Omni: Acknowledged. It keeps things simple from an implementation perspective, so keeping it.

Spearbit: Acknowledged.

5.2.2 `OmniBridgeL1._bridge`: `abi.encodeCall` is called redundantly during the "happy path"

Severity: Gas Optimization

Context: OmniBridgeL1.sol#L103, OmniBridgeL1.sol#L83, OmniBridgeL1.sol#L90

Description: The following line is being called twice during the execution of `OmniBridgeL1._bridge`:

```
abi.encodeCall(OmniBridgeNative.withdraw, (payor, to, amount, token.balanceOf(address(this)))),
```

The calculated value can be cached instead.

Recommendation: Consider writing an internal function - `_bridgeFee` that will be called inside `_bridge` instead of calling `bridge` and that will receive the value of `abi.encodeCall(...)` as a parameter instead. `bridgeFee` can be set to external as well which will be a bit more gas effective.

Omni: Fixed in commit ae390993.

Spearbit: Fix verified.

5.2.3 `_exec` can read parameter `xheader` directly from `calldata`

Severity: Gas Optimization

Context: OmniPortal.sol#L229

Description: The `xheader` parameter is only ever read in the `_exec` function. This can be read from `calldata` instead of copying to `memory`.

Recommendation: Change this line to:

```
function _exec(XTypes.BlockHeader calldata xheader, XTypes.Msg calldata xmsg_) internal {
```

- `forge s --diff`:

```
test_xsubmit_guzzle1_succeeds() (gas: -43 (-0.000%))
test_xsubmit_addValidator_succeeds() (gas: -43 (-0.000%))
test_xsubmit_reentrancy_reverts() (gas: -43 (-0.000%))
test_xsubmit_wrongStreamOffset_reverts() (gas: -507 (-0.001%))
test_xsubmit_wrongDestChainId_reverts() (gas: -507 (-0.001%))
test_exec_errorSize() (gas: -858 (-0.001%))
test_singleExec() (gas: -848 (-0.001%))
test_xsubmit_guzzle5_succeeds() (gas: -879 (-0.001%))
test_pauseAll() (gas: -880 (-0.001%))
test_xsubmit_xblock1_succeeds() (gas: -880 (-0.001%))
test_xsubmit_xblock1_chainB_succeeds() (gas: -880 (-0.001%))
test_xsubmit_addValidatorSet_succeeds() (gas: -923 (-0.001%))
test_xsubmit_notNewValSet_succeeds() (gas: -923 (-0.001%))
test_xsubmit_xblock2_succeeds() (gas: -1760 (-0.003%))
test_xsubmit_xblock2_chainB_succeeds() (gas: -1760 (-0.003%))
test_xsubmit_guzzle10_succeeds() (gas: -1971 (-0.003%))
test_xsubmit_guzzle25_succeeds() (gas: -5538 (-0.008%))
test_xsubmit_guzzle50_succeeds() (gas: -12470 (-0.017%))
test_exec_xmsg_succeeds() (gas: -40 (-0.026%))
test_exec_xmsgRevert_succeeds() (gas: -40 (-0.042%))
test_xcallToPortal__fails() (gas: -41 (-0.053%))
test_exec_behindOffset_reverts() (gas: -542 (-0.401%))
test_pause_unpause() (gas: -48107 (-0.495%))
test_upgrade() (gas: -70774 (-0.519%))
test_exec_aheadOffset_reverts() (gas: -502 (-2.056%))
test_exec_wrongDestChainId_reverts() (gas: -502 (-2.293%))
Overall gas change: -152261 (-0.009%)
```

Omni: Fixed in commit 7ad4c7ad.

Spearbit: Fix verified.

5.2.4 The return formula for `FeeOracleV1.feeFor` can be simplified to use one less multiplication

Severity: Gas Optimization

Context: FeeOracleV1.sol#L79

Description: The formula for can be simplified to use one less multiplication.

Recommendation: Use the following formula instead:

```
return protocolFee + (baseGasLimit + gasLimit) * execGasPrice + (dataGas * dataGasPrice);
```

- `forge s --diff`:

```
test_pauseAll() (gas: -53 (-0.000%))
test_xcall_gasLimitTooLow_reverts() (gas: -53 (-0.081%))
test_xcall_gasLimitTooHigh_reverts() (gas: -53 (-0.081%))
test_xcall_sameChain_reverts() (gas: -53 (-0.086%))
test_pauseXCall() (gas: -265 (-0.098%))
test_xcall_succeeds() (gas: -106 (-0.099%))
test_xcall_insufficientFee_reverts() (gas: -106 (-0.150%))
test_feeFor_succeeds() (gas: -106 (-0.222%))
test_feeFor() (gas: -477 (-0.395%))
Overall gas change: -1272 (-0.000%)
```

Omni: Fixed in commit 0f8ed51a.

Spearbit: Fixed.

5.2.5 The previous signature tuple can be read from `calldata` directly

Severity: Gas Optimization

Context: Quorum.sol#L36

Description: The previous signature tuple can be read from `calldata` directly.

Recommendation: Change this line to:

```
XTypes.SigTuple calldata prev = sigs[i - 1];
```

- `forge s --diff`:

```
test_xsubmit_noQuorum_reverts() (gas: -363 (-0.001%))
test_xsubmit_duplicateValidator_reverts() (gas: -518 (-0.001%))
test_xsubmit_guzzle1_succeeds() (gas: -729 (-0.001%))
test_xsubmit_reentrancy_reverts() (gas: -729 (-0.001%))
test_xsubmit_guzzle5_succeeds() (gas: -737 (-0.001%))
test_xsubmit_invalidAttestationRoot_reverts() (gas: -732 (-0.001%))
test_xsubmit_invalidMsgs_reverts() (gas: -732 (-0.001%))
test_xsubmit_wrongStreamOffset_reverts() (gas: -733 (-0.001%))
test_xsubmit_wrongDestChainId_reverts() (gas: -733 (-0.001%))
test_pauseAll() (gas: -737 (-0.001%))
test_xsubmit_xblock1_succeeds() (gas: -737 (-0.001%))
test_xsubmit_guzzle10_succeeds() (gas: -748 (-0.001%))
test_xsubmit_xblock1_chainB_succeeds() (gas: -738 (-0.001%))
test_xsubmit_guzzle25_succeeds() (gas: -779 (-0.001%))
test_xsubmit_guzzle50_succeeds() (gas: -832 (-0.001%))
test_xsubmit_addValidator_succeeds() (gas: -849 (-0.001%))
test_xsubmit_xblock2_succeeds() (gas: -1474 (-0.002%))
test_xsubmit_xblock2_chainB_succeeds() (gas: -1476 (-0.002%))
test_xsubmit_addValidatorSet_succeeds() (gas: -1586 (-0.002%))
test_xsubmit_notNewValSet_succeeds() (gas: -1586 (-0.002%))
test_verify_powerBasedQourum_succeeds() (gas: -363 (-0.046%))
test_verify_noQuorum_fails() (gas: -363 (-0.127%))
test_verify_sigsNotSorted_reverts() (gas: -362 (-0.128%))
test_verify_duplicateValidator_reverts() (gas: -521 (-0.184%))
test_verify_largeValset_succeeds() (gas: -24729 (-0.226%))
test_verify_succeeds() (gas: -725 (-0.245%))
test_pause_unpause() (gas: -87831 (-0.904%))
test_upgrade() (gas: -131748 (-0.966%))
Overall gas change: -264190 (-0.004%)
```

Omni: Fixed in commit d07b61b7.

Spearbit: Fixed.

5.3 Informational

5.3.1 Redundant `require` statement in `Quorum.verify`

Severity: Informational

Context: Quorum.sol#L37-L38

Description: In this context the first `require` statement is redundant since the second one checks for the monotonicity of the `sig.validatorAddr`.

Recommendation: The following can be removed:

```
require(sig.validatorAddr != prev.validatorAddr, "Quorum: duplicate validator");
```

Omni: Fixed in commit 82d794d0.

Spearbit: Fixed.

5.3.2 Each name space size is `1024`

Severity: Informational

Context: Predeploys.sol#L58-L61, Predeploys.sol#L9

Description: Each name space size is `1024` or 2^{10}. So when determine whether an address is a predeploy only shifting `10` bits would be enough.

Recommendation: Change `isPredeploy` to:

```
function isPredeploy(address addr) internal pure returns (bool) {
    return (uint160(addr) >> 10 == uint160(OmniNamespace) >> 10)
        || (uint160(addr) >> 10 == uint160(OctaneNamespace) >> 10);
}
```

Omni: Fixed in commit b121a383.

Spearbit: Fixed.

5.3.3 Missing events

Severity: Informational

Context: OmniBridgeNative.sol#L101, OmniBridgeNative.sol#L180-L182, PausableUpgradeable.sol#L33-L46, Staking.sol#L48, Staking.sol#L53, Staking.sol#L57, Staking.sol#L95-L122

Description: When the following storage parameters are changed no events are emitted:

- `Staking.{isAllowlistEnabled, isAllowedValidator}`.
- `OmniBridgeNative.{l1ChainId, omni, l1Bridge}`.
- In `OmniBridgeNative.withdraw` when the call to `to` fails and `claimable` is updated it might be useful to emit a custom event to indicate that.
- `PausableUpgradeable.PauseableStorage`.

Recommendation: For some off-chain tooling it might be useful to emit events when the parameters in this context are changed or other on-chain actions are performed where it would be desired to track off-chain.

Omni: New events have been added in commit 769f1de0.

Note: not adding explicit event for claimable in bridge withdraw -- we can use the `success` field in the `Withdraw` event.

Spearbit: The point about `OmniBridgeNative.withdraw` has not been addressed.

5.3.4 Storage slots are allocated during genesis and `initialize` would not be callable

Severity: Informational

Context: OmniBridgeNative.sol#L15-L20, OmniBridgeNative.sol#L76-L78, PortalRegistry.sol#L10-L15, PortalRegistry.sol#L60-L62, Staking.sol#L11-L17, Staking.sol#L55-L58, Upgrade.sol#L11-L15, Upgrade.sol#L43-L45

Description: Sine the storage slots are allocated during genesis according to the NatSpec:

```
/**
 * @dev This contract is predeployed, and requires storage slots to be set in genesis.
 *      Genesis storage slots must:
 *          - set _owner on proxy
 *          - set _initialized on proxy to 1, to disable the initializer
 *          - set _initialized on implementation to type(uint64).max, to disabled all initializers
 *          ...
```

Then the function `initialize` would not be able to be called.

Recommendation: Either the NatSpec should be updated or the `initialize` be removed. If not removed a comment regarding why it is still present would be useful.

Omni: NatSpec is updated to reflect the setup strategy in 2906ec14.

Spearbit: Fixed.

5.3.5 `claim` function can be invoked on `ConfLevel.Latest`

Severity: Informational

Context: OmniBridgeNative.sol#L150

Description: `OmniBridgeNative.claim` is callable through an `xcall` of level `ConfLevel.Latest`. This function can be called zero, or multiple times upon `xcall` invocation. This isn't inherently vulnerable even if the function gets called multiple times - the value is only claimed the first time. But it may be a surprise when submitting an xcall that the `claim` never goes through.

Recommendation: Ensure documentation states that this is the case when attempting to `claim` claimable tokens across the bridge using `ConfLevel.Latest`. Alternatively, require that this function is only called on `ConfLevel.Finalized`.

Omni: Acknowledged. We allow `claim` to be called via `ConfLevel.Latest` because all information on what can be claimed lives on the native bridge contract, so the xmsg need only arrive to be authorized.

Spearbi: Acknowledged.

5.3.6 `OmniPortal.sol`: Lower than expected number of available valsets

Severity: Informational

Context: *(No context files were provided by the reviewer)*

Description: The `_minValSet` function in the `OmniPortal` contract specifies a limit such that only `XSubValsetCutoff` number of accepted valsets are available at any point of time. However, fewer than that can be available due to the nature of the `_addValidatorSet` function.

In the `_addValidatorSet` function, the consensus can pass in any `valSetId` and the only requirement is that the passed in `valSetId` has not been used up before. However, there is no constraint that the consensus uses consecutive `valSetId` values, and can effectively skip over certain `valSetId` values, leaving them unset.

Say the `latestValSetId` is 10 and `xsubValsetCutoff` is 5. Then according to `_minValSet`, only the values in index 6 through 10 (5 values) are acceptable. However, now an update can set a list of validators for `valSetId=14`. Now, only indices 10 through 14 are available according to `_minValSet`, and since 11,12,13 were never set, effectively there are only 2 available validator sets available now (index 10 and index 15).

So there can be lower than `XSubValsetCutoff` number of usable validator sets available at any given point.

Recommendation: In the `_addValidatorSet` function, consider only setting the validators sequentially and updating `valSetId` by 1. This way indices cannot be skipped, reducing the effective number of usable validator sets.

Omni: Acknowledged. The consensus chain enforces incrementing validator set ids.

Spearbit: Acknowledged.

5.3.7 `OmniBridgeL1._bridge`: Inconsistency in the checks against `msg.value`

Severity: Informational

Context: OmniPortal.sol#L140, OmniBridgeL1.sol#L83

Description: During the flow of `OmniBridgeL1._bridge` msg.value is checked twice, once inside `OmniBridgeL1._bridge`:

```
require(msg.value == bridgeFee(payor, to, amount), "OmniBridge: incorrect fee");
```

and the other inside `OmniPortal.xcall`:

```
uint256 fee = feeFor(destChainId, data, gasLimit);
require(msg.value >= fee, "OmniPortal: insufficient fee");
```

As we can see the first check uses == while the second uses >=.

Recommendation: Consider changing the first expression to use >= instead which will allow small flexibility in case of fee fluctuation.

Omni: Acknowledged. We made this choice in the bridge to save users fees (in some cases). The bridge check on our ui is synchronous before submitting bridge transaction and we can save people a little money. It could fail yes, but would also fail in cause where fee is too low with equal probability.

Spearbit: Acknowledged.

5.3.8 `OmniPortal.initialize`: Inconsistency of event emissions for `inXMsgOffset` and `inXBlockOffset`

Severity: Informational

Context: OmniPortal.sol#L104-L105

Description: Both `inXMsgOffset` and `inXBlockOffset` are being set in the `initialize` function without any event emissions unlike in `setInXMsgOffset` and `setInXBlockOffset` respectively.

Recommendation: Consider implementing internal functions that will be called for the two code paths and that will both set the value and emit an event.

Omni: Fixed in commit ae390993.

Spearbit: Fix verified.

5.3.9 `OmniBridgeNative`: `setup` might be redundant adding additional unnecessary trust assumptions

Severity: Informational

Context: OmniBridgeNative.sol#L179, OmniBridgeNative.sol#L76-L78

Description: `OmniBridgeNative` is supposed to be deployed as an upgradeable contract which will be initialize by calling `initialize` and then `setup`. `initialize` can be called only once as it should but it is not the case with `setup` which can be called more than once by the owner of the contract. In case there is no need to call `setup` more than once, having this logic as a separate function might be redundant adding additional unnecessary trust assumptions on the owner of the contract.

Recommendation: Consider merging the content of the `setup` function into `initialize` and remove this function.

Omni: We use `setup` because we do not know these values before hand. Though, we kinda do -- we use create3 deployments so can state what the addresses will be. Though rather than adding more inputs to predeploy allocation script, we opted to do this setup post deployment.

Spearbit: Acknowledged.

5.3.10 The NatSpec comment for `OmniBridgeNative` regarding the genesis storage slots are not entirely accurate

Severity: Informational

Context: OmniBridgeNative.sol#L15-L20, OmniBridgeNative.sol#L76-L78

Description: Disabling all initialisers are missing in `OmniBridgeNative` deployed implementation contract. Since there is also no explicit `constructor` function. The following would not be needed in the NatSpec `@dev` comment is true:

```
constructor() {
   _disableInitializers();
}
```

but then if the storage slots are set in genesis, then the included `initialize` function is not necessary:

```
function initialize(address owner_) external initializer {
   __Ownable_init(owner_);
}
```

In `./contracts/allocs/mainnet.json` (the genesis's allocation file) we have:

```
  "0x121e240000000000000000000000000000000002": {
    "balance": "0x52b7d2dcc80cd2e4000000",
    "code": "0x608060405261000c61000e565b005b7f...",
    "nonce": "0x0",
    "storage": {
      "0x360894a13ba1a3210667c828492db98dca3e2076cc3735a920a3ca505d382bbc":
↪   "0x0000000000000000000000ede1dbffffffffffffffffffffffffffffffffd",
      "0x9016d09d72d40fdae2fd8ceac6b6234c7706214fd39c1cd1e609a0528c199300":
↪   "0x0000000000000000000000000000000000000000000000000000000000000dead",
      "0xb53127684a568b3173ae13b9f8a6016e243e63b6e8ee1178d6a717850b5d6103":
↪   "0x0000000000000000000004ca2f0e23e0d0a5b325e0511bc9adf36f0c743c5",
      "0xf0c57e16840df040f15088dc2f81fe391c3923bec73e23a9662efc9c229c6a00":
↪   "0x0000000000000000000000000000000000000000000000000000000000000001"
    }
  },
  // ...
  "0xede1dbffffffffffffffffffffffffffffffffd": {
    "balance": "0x0",
    "code": "0x6080604052600436106101405760003560e...",
    "nonce": "0x0",
    "storage": {
      "0xf0c57e16840df040f15088dc2f81fe391c3923bec73e23a9662efc9c229c6a00":
↪   "0x0000000000000000000000000000000000000000000000000fffffffffffffff"
    }
  },
"0x4ca2f0e23e0d0a5b325e0511bc9adf36f0c743c5": {
    "balance": "0x0",
    "code": "0x608060405260043610610041a5760003560e01...",
    "nonce": "0x1",
    "storage": {
      "0x0000000000000000000000000000000000000000000000000000000000000000":
↪   "0x0000000000000000000000000000000000000000000000000000000000000dead"
    }
  },
```

or:

```
OmniBridgeNative (proxy)        : 0x121e2400000000000000000000000000000002
IMPLEMENTATION_SLOT             : 0xede1dbffffffffffffffffffffffffffffffffd
OwnableStorageLocation          : 0x0000000000000000000000000000000000dead << here
ERC1967Utils.ADMIN_SLOT         : 0x4ca2f0e23e0d0a5b325e0511bc9adf36f0c743c5
INITIALIZABLE_STORAGE           : 0x01 ({_initialized: 1, _initializing: 0})

Native Token Balance: 10e26
```

```
OmniBridgeNative (implementation): 0xede1dbffffffffffffffffffffffffffffffffd
INITIALIZABLE_STORAGE            : 0xffffffffffffffff ({_initialized: 0xffffffffffffffff,
↪  _initializing: 0})
```

```
Proxy Admin                     : 0x4ca2f0e23e0d0a5b325e0511bc9adf36f0c743c5
SLOT_0x0                        : 0x0000000000000000000000000000000000dead <<
```

Note that with the current allocation `JSON` files for different network, the `initialize` would not be callable on the `TUP` proxy since the value of `_initialized` is already `1` on genesis.

Recommendation: Make sure the comment or code are modified to match accordingly.

Omni: The NatSpec comments are updated to reflect the setup strategy in commit 2906ec14:

```
/**
 ...
 *      initialize(...) is called pre-deployment, in sctips/genesis/AllocPredeploys.s.sol
 *      iniitializers on the implementation are disabled via manual storage updates, rather than in a
↪  constructor.
 *      If an new implementation is required, a constructor should be added.
```

Spearbit: Fixed.

5.3.11 Redundant `override` keyword

Severity: Informational

Context: WOmni.sol#L47, WOmni.sol#L52

Description: When overriding an interface function, the `override` keyword is not necessary.

Recommendation: The `override` keyword can be removed.

Omni: Fixed in commit 4620a253.

Spearbit: Fixed.

5.3.12 Variable name shadowing

Severity: Informational

Context: OmniPortal.sol#L369

Description:

- OmniPortal.sol#L369: `valSet` has been shadowed here.

Recommendation: Consider renaming the variable to avoid shadowing.

Omni: Fixed in commit ebe2103e.

Spearbit: Fixed.

5.3.13 Typos

Severity: Informational

Context: IOmniPortal.sol#L39-L40, IOmniPortal.sol#L51, OmniBridgeNative.sol#L19, OmniBridgeNative.sol#L72, OmniPortal.sol#L34, OmniPortalConstants.sol#L35, OmniPortalStorage.sol#L80-L81, OmniPortalStorage.sol#L86, PortalRegistry.sol#L72, Quorum.sol#L8, XBlockMerkleProof.sol#L26, XTypes.sol#L45

- XBlockMerkleProof.sol#L26: should be `against`.
- PortalRegistry.sol#L72: comment refers to `PortalRegistry.get(..)` method and needs to be updated for this `list()` function.
- OmniPortalConstants.sol#L35: `xmg` → `xmsg`.
- OmniPortal.sol#L34: grammar - `Modifier that requires that an action ....`
- OmniPortalStorage.sol#L86: `shardIdj` → `shardId`.
- XTypes.sol#L45: copy and pasted field name `sourceChainId` → `consensusChainId`
- OmniPortalStorage.sol#L80-L81: the NatSpec comment for `inXMsgOffset` is copied from `outXMsgOffset` and should be corrected.
- IOmniPortal.sol#L51: `error` is a special keyword in `Solidity`. It would be best to use a different parameter name. Also make sure to update the NatSpec.
- IOmniPortal.sol#L39-L40: the NatSpec params `success` and `relayer` are swapped in order with the actual event. They should be swapped to be accurate.
- OmniBridgeNative.sol#L72: should be `fails`.
- OmniBridgeNative.sol#L19: `_initialized` is of type `uint64` so we need to change `255` to `0xffffffffffffffff`.
- Quorum.sol#L8: `Quorom` → `Quorum`.

Recommendation: Consider fixing the typos to improve readability and maintainability of the codebase.

Omni: Fixed in PR 1883.

Spearbit: Looks good except one typo has been introduced in file r1782825881.

Omni: Typo fixed in commit 32d88893.

Spearbit: Fixed.