Quantstamp Security Assessment Certificate

June 9th 2022 — Quantstamp Verified

Nomad

This audit report was prepared by Quantstamp, the leader in blockchain security.

Executive Summary

Type	Bridge
Auditors	Souhail Mssassi, Research Engineer Andy Lin, Senior Auditing Engineer Alejandro Padilla Gaeta, Research Engineer
Timeline	2022-04-12 through 2022-06-06
EVM	London
Languages	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	Nomad Docs
Documentation Quality	Medium
Test Quality	Medium

Source Code

Repository	Commit
monorepo	17f0557
zodiac-module-nomad	d887024
zodiac-module-nomad	482c1e2
zodiac-module-nomad	93daaf9

Total Issues	**40** (16 Resolved)
High Risk Issues	**1** (1 Resolved)
Medium Risk Issues	**6** (6 Resolved)
Low Risk Issues	**18** (7 Resolved)
Informational Risk Issues	**9** (1 Resolved)
Undetermined Risk Issues	**6** (1 Resolved)

High Risk	The issue puts a large number of users' sensitive information at risk, or is reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
Medium Risk	The issue puts a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or is reasonably likely to lead to moderate financial impact.
Low Risk	The risk is relatively small and could not be exploited on a recurring basis, or is a risk that the client has indicated is low-impact in view of the client's business circumstances.
Informational	The issue does not post an immediate risk, but is relevant to security best practices or Defence in Depth.
Undetermined	The impact of the issue is uncertain.
Unresolved	Acknowledged the existence of the risk, and decided to accept it without engaging in special efforts to control it.
Acknowledged	The issue remains in the code but is a result of an intentional business or design decision. As such, it is supposed to be addressed outside the programmatic means, such as: 1) comments, documentation, README, FAQ; 2) business processes; 3) analyses showing that the issue shall have no negative consequences in practice (e.g., gas analysis, deployment settings).
Fixed	Adjusted program implementation, requirements or constraints to eliminate the risk.
Mitigated	Implemented actions to minimize the impact or likelihood of the risk.

Summary of Findings

Initial audit:
Through reviewing the code, we found **40 potential issues** of various levels of severity:1 high-severity, 6 medium-severity, 18 low-severity, 9 informational-security and 6 undermined issues. We recommend addressing all the issues before deploying the code.
After the re-audit:
The Nomad team has fixed the majority of the issues and the contracts are ready to be deployed, we also found a new issue the QSP-40.

ID	Description	Severity	Status
QSP-1	Impossibility To Recover From A Failed State	^ Medium	Fixed
QSP-2	Messages Can Be Proven And Processed In The Replica Even On `Failed` State	^^ High	Fixed
QSP-3	Messages Can Be Delivered Out Of Order	^ Medium	Fixed
QSP-4	Long Queues Can Lead To Denial Of Service	^ Medium	Mitigated
QSP-5	`_dust` Method Can Be Abused	v Low	Acknowledged
QSP-6	Watcher Availability Concern	^ Medium	Mitigated
QSP-7	Replay Attack In The `improperUpdate`	v Low	Acknowledged
QSP-8	Missing Verification On The `FEE_NUMERATOR`	^ Medium	Fixed
QSP-9	Missing Validation On The `optimisticSeconds`	^ Medium	Fixed
QSP-10	Liquidity Provider Fee Can Be Bypassed	v Low	Fixed
QSP-11	Front Running Can Make Liquidity Providers Lose Their Funds	v Low	Fixed
QSP-12	BridgeToken Allows The Owner Of The Contract To Burn Tokens Even Without Their Approval	v Low	Acknowledged
QSP-13	`_sendToAllRemoteRouters` Will Not Complain If The Remote Router Is Not Present	O Informational	Acknowledged
QSP-14	Possible To Change `BridgeToken` Decimals	v Low	Acknowledged
QSP-15	Contracts Might Not Be Transferred To Rightful Owner	v Low	Mitigated
QSP-16	Custom Tokens Could Be Malicious	v Low	Acknowledged
QSP-17	Possible To Renounce Ownership	v Low	Mitigated
QSP-18	Signature Replay Attack: Multiple Deployments	v Low	Acknowledged
QSP-19	Proving With An Empty Leaf	v Low	Acknowledged
QSP-20	Integer Overflow /Underflow	v Low	Fixed
QSP-21	Tighten Validations Around Message	v Low	Acknowledged
QSP-22	Abusing Underflow	v Low	Mitigated
QSP-23	Missing Input Validation	v Low	Acknowledged
QSP-24	Incompatibility With Deflationary Tokens	v Low	Acknowledged
QSP-25	Approve Race Condition	v Low	Acknowledged
QSP-26	Floating Pragma	v Low	Fixed
QSP-27	Modifier Allows This Contract To Be The Caller	O Informational	Acknowledged
QSP-28	BridgeRouter Must Be The Owner Of The TokenRegistry To Work	O Informational	Mitigated
QSP-29	Signature Replay Attack: Hard Fork	O Informational	Acknowledged
QSP-30	Mass Un-enrollment	O Informational	Acknowledged
QSP-31	Susceptible To Signature Malleability	O Informational	Acknowledged
QSP-32	Missing Events	O Informational	Acknowledged
QSP-33	Trusted Actors Risk And External Agents Have Too Much Power	O Informational	Acknowledged
QSP-34	Transactions Are Marked As Processed Even If They Fail	? Undetermined	Fixed
QSP-35	Upgradeable Contracts Can Be Working With Older Versions	? Undetermined	Acknowledged
QSP-36	No Enforcement On The Governance Messages To Be Delivered	? Undetermined	Acknowledged
QSP-37	Possible To Run `executeCallBatch` While Local Governance Router Is On Recovery	? Undetermined	Acknowledged
QSP-38	Updater Stop Signing New Roots	? Undetermined	Acknowledged
QSP-39	Diverge In The Updater During Updater Rotation	? Undetermined	Acknowledged
QSP-40	Initializer Not Disabled On Implementation Contract	O Informational	Acknowledged

Quantstamp Audit Breakdown

Quantstamp's objective was to evaluate the repository for security-related issues, code quality, and adherence to specification and best practices.

Possible issues we looked for included (but are not limited to):

- Transaction-ordering dependence
- Timestamp dependence
- Mishandled exceptions and call stack limits
- Unsafe external calls
- Integer overflow / underflow
- Number rounding errors
- Reentrancy and cross-function vulnerabilities
- Denial of service / logical oversights
- Access control
- Centralization of power
- Business logic contradicting the specification
- Code clones, functionality duplication
- Gas usage
- Arbitrary token minting

Methodology

The Quantstamp auditing process follows a routine series of steps:

1. Code review that includes the following
   i. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   ii. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   iii. Comparison to specification, which is the process of checking whether the code does what the specifications, sources, and instructions provided to Quantstamp describe.
2. Testing and automated analysis that includes the following:
   i. Test coverage analysis, which is the process of determining whether the test cases are actually covering the code and how much code is exercised when we run those test cases.
   ii. Symbolic execution, which is analyzing a program to determine what inputs cause each part of a program to execute.
3. Best practices review, which is a review of the smart contracts to improve efficiency, effectiveness, clarify, maintainability, security, and control based on the established industry and academic practices, recommendations, and research.
4. Specific, itemized, and actionable recommendations to help you take steps to secure your smart contracts.

Toolset

The notes below outline the setup and steps performed in the process of this audit.

Setup

Tool Setup:

- Slither v0.8.2

Steps taken to run the tools:

1. Install the Slither tool: `pip3 install slither-analyzer`
2. Run Slither from the project directory: `slither .`

Findings

QSP-1 Impossibility To Recover From A Failed State

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `Home.sol`, `NomadBase.sol`, `Replica.sol`

Description: Whenever fraud is detected in the `Home` or `Replica` contracts, they will be moved to a `Failed` state (via the `doubleUpdate` or `improperUpdate` methods). However, once a contract moves to that state, it's not possible to return it to `Active`. The only way out is to upgrade the contracts' code or deploy new instances.

Recommendation: Design a mechanism for governance or a privileged account to get the system back to `Active` state once fraud has been dealt with.

Update: The Nomad team fixed the issue. Given this issue could also have been solved through a contract upgrade, we decided to move its severity to `Medium`.

QSP-2 Messages Can Be Proven And Processed In The Replica Even On `Failed` State

Severity: *High Risk*

Status: Fixed

File(s) affected: `Replica.sol`

Description: The `prove`, `process`, and `proveAndProcess` methods can be called even if the `Replica` contract is on `Failed` state. Therefore, if governance does not deal with the replica immediately (replace replica in the corresponding `XAppConnectionManager` or remove fraudulent roots via the `setConfirmation` method), fraudulent roots might have enough time to be confirmed, thereby letting fraudulent messages to be proven and processed.

Recommendation: Annotate the `prove`, `process`, and `proveAndProcess` with the `notFailed` modifier to ensure that they cannot be executed while the contract is on `Failed` state.

Update: The Nomad team is moving to a design where the `FAILED` state on the replica is superfluous, and now the applications have the responsibility of blocking replicas when they observe fraud. We recommend describing this extensively in your documentation for external developers.

QSP-3 Messages Can Be Delivered Out Of Order

Severity: Medium Risk

Status: Fixed

File(s) affected: `Replica.sol`

Description: According to the Nomad documentation, the replica has to ensure that messages are delivered in order. However, that's not the case. As long as a message has been marked as `PROVEN`, the processor is free to call the `process` method in the replica in any order it wants. This is a problem because a malicious processor or malicious third-party (the `process` method is not protected, so anyone could call it) could force the replica to process their messages in a way that is beneficial to them (front-run).

Recommendation: We strongly recommend adding code to the replica to ensure that messages are always processed in order. Otherwise, if the risk is deemed acceptable, update the Nomad documentation to highlight that the replica can deliver messages out of order and that it's up to the dApps to protect themselves against this.

Update: The Nomad team pointed out that their original documentation was wrong; messages cannot be delivered in order as that could be a dangerous DOS attack vector. Therefore, they updated the documentation to properly reflect this.

QSP-4 Long Queues Can Lead To Denial Of Service

Severity: Medium Risk

Status: Mitigated

File(s) affected: `Home.sol`, `Queue.sol`

Description: Whenever the `update` method in the `Home` contract is called, it will validate if the queue contains the new root. However, the queue `contains` method does a linear scan of all the items in the queue. If the root has not been updated in a long time (either mistakenly or maliciously), the queue might have too many items, causing the `contains` method to throw an `Out of Gas` exception.

Recommendation: Consider if a mapping or a separate data structure could be used to make the `contains` method more resilient to `Out of Gas` exceptions.

Update: The Nomad team acknowledges the issue and has decided to mitigate the issue by having an Updater sign a root near the beginning of the queue, thereby preventing the `Out of Gas` exception; the Updater will have to repeat the same steps until the queue is empty. In the future, they will solve the problem as part of a new design. Meanwhile, we still recommend documenting the mitigation steps so that the Updater is well aware of what to do if this problem occurs.

QSP-5 `_dust` Method Can Be Abused

Severity: Low Risk

Status: Acknowledged

File(s) affected: `BridgeRouter.sol`

Description: The `_dust` method is used to send recipients a substantial amount of `Ether` for gas bootstrapping (0.06 `Ether`, which is around 140 USD currently). The method only checks that the recipient ETH balance is less than the `DUST_AMOUNT`. A malicious trader could repetitively send small trades, to constantly receive the `DUST_AMOUNT` and drain the `BridgeRouter` from Ether.

Recommendation: Consider if the `_dust` functionality is needed at all. If not, remove it. Otherwise, add additional checks to make sure it cannot be abused.

Update: The Nomad team is aware that this function can be abused, but believes that the risk is low as it will never contain big amounts of Ether.

QSP-6 Watcher Availability Concern

Severity: Medium Risk

Status: Mitigated

File(s) affected: `ethereum.ts`, `moonbeam.ts`

Description: Nomad architecture assumes the liveness of the watcher running on top of it to secure the Dapp running on top of it. We need the watcher to unroll from the replica within 30 minutes. Therefore, the availability of the watcher is critical. The current deployment is only configured with a single watcher, which concerns the availability of any hardware incident that occurs.

Recommendation: Deploy and configure with a minimum of 2 watchers, ideally cross-region, for high availability

Update: The Nomad team highlighted that there are multiple watcher instances (sharing the same key), deployed across different regions, to protect against cloud provider failure.

QSP-7 Replay Attack In The `improperUpdate`

Severity: Low Risk

Status: Acknowledged

File(s) affected: `BridgeRouter.sol`, `Replica.sol`

Description: In the `improperUpdate` function, we are taking the `_signature` as a parameter to verify the correctness of the transaction. The issue here is that the signature is not associated to any nonce or address. Thus, the signature can be replayed since there is no nonce associated.

Recommendation: Consider adding a nonce to each signature.

Update: The Nomad team believes that the likelihood of this occuring in practice is quite low.

QSP-8 Missing Verification On The `FEE_NUMERATOR`

Severity: *Medium Risk*

Status: Fixed

File(s) affected: `BridgeRouter.sol`

Description: During an upgrade of the contract, `PRE_FILL_FEE_NUMERATOR` could be set to a value greater than 10000. This behaviour would result in impacting the logic of the contract.

Recommendation: To solve the issue, a check should be placed in the function that makes sure that the Percentage is always less than 100%.

Update: The Nomad team deprecated the fast liquidity functionality and removed most of the code. This issue cannot occur anymore.

QSP-9 Missing Validation On The `optimisticSeconds`

Severity: *Medium Risk*

Status: Fixed

Description: The `optimisticSeconds` parameter is not validated, so it could be set to 0 or a minimal amount. This would allow updates to be accepted in a very short time, thereby not leaving the replica with enough time to respond to fraud (this is also a problem in the `initialize` method).

Recommendation: Consider verifying optimisticSeconds to be greater than a reasonable minimum value.

Update: The Nomad team added validation to ensure that the `optimisticSeconds` value cannot be too low or too high. However, this check can be bypassed to set low values in test environments; therefore, we recommend adding checks to the deployment script to ensure it's not possible to set low values in production by mistake.

QSP-10 Liquidity Provider Fee Can Be Bypassed

Severity: *Low Risk*

Status: Fixed

File(s) affected: `BridgeRouter.sol`

Description: In the `_applyPreFillFee` function, the `_amtAfterFee` is calculated using the following formula

```
(_amnt * PRE_FILL_FEE_NUMERATOR) /
PRE_FILL_FEE_DENOMINATOR;
```

The issue here is that is if `_amt * PRE_FILL_NUMERATOR` is less than `PRE_FILL_FEE_DENOMATOR` the `_amtAfterFee` will be equal to 0.

Recommendation: Consider verifying that `_amt * PRE_FILL_NUMERATOR` is always greater than `PRE_FILL_FEE_DENOMATOR`.

Update: The Nomad team deprecated the fast liquidity functionality and removed most of the code. This issue cannot occur anymore.

QSP-11 Front Running Can Make Liquidity Providers Lose Their Funds

Severity: *Low Risk*

Status: Fixed

File(s) affected: `BridgeRouter.sol`

Description: The `preFill` method allows a liquidity provider (LP) to fill a transfer that hasn't been verified yet for a fee. This method ensures that only one liquidity provider can provide liquidity to a single transfer (`L230`). However, it does not verify if the transfer has already been processed by the `handle` method (`L107`). A malicious processor (or even a malicious miner) could wait to send the message that will trigger the `handle` method just before a `preFill` method call is done, thereby letting the recipient receive funds from both, the router and the liquidity provider. To make things worse, once the `send` method has been processed, another LP would be able to call `preFill` again for the same transaction.

Recommendation: Consider modifying the `handle` method in the `BridgeRouter` contract to always set the `liquidityProvider` for the transaction if the `fastEnabled` flag is on. If the transaction was not pre-filled by an LP, set the router itself as the liquidity provider. Otherwise, leave the `liquidityProvider` already set (remove the `delete` statement in `L337`).

Update: The Nomad team deprecated the fast liquidity functionality and removed most of the code. This issue cannot occur anymore. However, some of the fast liquidity code is still present (for example, the `_handleTransfer` still has cases for fast liquidity). We recommend removing all dead code to avoid any potential bug.

QSP-12 BridgeToken Allows The Owner Of The Contract To Burn Tokens Even Without Their Approval

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `BridgeToken.sol`

Description: The owner of a `BridgeToken` instance (`BridgeRouter`) can burn tokens of any address, even without the token owner's consent. While the router is unlikely to steal tokens, future iterations of the router could mistakenly burn tokens without the user consent.

Recommendation: Modify the `burn` method in the `BridgeToken` contract to only be able to destroy tokens that the owning address has approved (via the `approve` methods). We recommend following the same approach used by OpenZeppelin in their `ERC20Burnable contract`.

Update: The Nomad team confirmed that this is by design. Moreover, the risk is quite low as the owner of the `BridgeToken` is the `BridgeRouter` itself.

QSP-13 `_sendToAllRemoteRouters` Will Not Complain If The Remote Router Is Not Present

Severity: *Informational*

Status: Acknowledged

File(s) affected: `GovernanceRouter.sol`

Description: When transferring the governor, the `GovernanceRouter` will call the `_sendToAllRemoteRouters` method to send the message to all remote chains. However, the method does not validate if a domain has no router configured (empty router). If the domain has no router, the method will still dispatch the message without any indication that the method will fail on the receiving domain.

Recommendation: Add a validation with a proper error message on `_sendToAllRemoteRouters` to make sure that a router is available before dispatching the message to the remote chain.

Update: The Nomad team has decided not to address this issue, as the impact is quite low; we agree with their assessment.

QSP-14 Possible To Change `BridgeToken` Decimals

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `BridgeToken.sol`, `OZERC20.sol`

Description: The `setDetails` method allows anyone to change the token information (name, symbol, and decimals) as long as the new details match the `detailsHash` set by the owner. This means that, with the owner's consent, it's possible to change the decimals of the token. If mishandled (mistakenly or maliciously), this could have serious consequences, as the decimals are needed to perform proper token math.

Recommendation: Consider if it's necessary to ever allow changing decimals after they have been set. If it's still deemed necessary, document that it is discouraged to ever introduce changes to them, as that can have unforeseen consequences.

Update: The ability to change decimals is important on the overall design. Given the impact is also low, the Nomad team decided not to make any code changes.

QSP-15 Contracts Might Not Be Transferred To Rightful Owner

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `GnomadModule.sol`, `BridgeToken.sol`, `NomadBase.sol`, `UpdaterManager.sol`, `XAppConnectionManager.sol`, `UpgradeBeaconController.sol`, `XAppConnectionClient.sol`

Description: Contracts inheriting from `Ownable` or `OwnableUpgradeable` will have the address that deployed the contract as the owner. If the deployment script does not change the ownership right away, the deployment address will keep ownership (along with all its privileges) instead of the rightful owner (like governance).

Recommendation: Consider transferring ownership of the contracts to the rightful owner directly in the constructor (or initializer function).

Update: The Nomad team added checks in the deployment scripts to ensure that the contracts end up assigned to the right owner.

QSP-16 Custom Tokens Could Be Malicious

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `BridgeRouter.sol`

Description: Custom ERC20 tokens might not work as expected due to faulty or malicious implementations (for example, not transferring tokens or burning tokens as expected). **Note:** This is marked low, as only the owner can enrol custom ERC20 tokens.

Recommendation: Make sure to verify that the code of custom ERC20 tokens is correct and is not malicious. If it's not possible to do due diligence on every custom token, consider adding checks in code to ensure that the balance after the operations on custom ERC20 tokens matches what the router expects.

Update: The Nomad team believes that governance vetting of tokens should be sufficient to prevent malicious tokens from being enrolled.

QSP-17 Possible To Renounce Ownership

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `GnomadModule.sol`, `BridgeToken.sol`, `NomadBase.sol`, `UpdaterManager.sol`, `XAppConnectionManager.sol`, `UpgradeBeaconController.sol`, `UpgradeBeaconController.sol`

Description: All the contracts that extend from `Ownable` or `OwnableUpgradeable` inherit a method called `renounceOwnership`. The owner of the contract can use this method to give up their ownership, thereby leaving the contract without an owner. If that were to happen, it would not be possible to perform any owner-specific functionality on that contract any more.

Recommendation: Consider overriding the `renounceOwnership` function so that ownership cannot be renounced.

Update: The Nomad team fixed the issue in most of the files by overriding the `renounceOwnership` as a no-op to make sure that ownership cannot be renounced. However, the fix was not introduced to the `UpgradeBeaconController` file. We recommend adding the same fix if ownership of that file should not be renounced.

QSP-18 Signature Replay Attack: Multiple Deployments

Severity: *Low Risk*

Status: Acknowledged

Description: The attacker can replay the message's signature if there are multiple deployments in the same chain running with the same updater. In the worst case, one can use it to call `Home.sol:improperUpdate` to fail the home contract.

Exploit Scenario: 1. The team deploys `Home.sol` and has some messages dispatched.

1. The Updater signs a root and calls `Home.sol:update` with signature `sign1` that moves the root from an empty one (the initial commit root) to `root1`.
2. After some period, the team decides to deploy another `Home.sol` (e.g., For a different app to decouple or other projects). Nonetheless, the team deployed with the same updater.
3. Anyone can call the `Home.sol:improperUpdate` immediately after the second deployment with the signature `sign1` from the first deployment. Since the initial commit root should be the same for all home deployments (null Merkle tree root), it will pass the verification.

4. Now, the status of the home contract has become FAILED.

Recommendation: Consider following the EIP712 suggestion that includes: `name`, `version`, `chainId`, `verifyingContract`, and `salt` in the domain separator (link). Current implementation inside `NomadBase.sol:_homeDomainHash` function only includes `_homeDomain` and the name "NOMAD". We recommend adding `version`, `verifyingContract` and `salt` to the domain separator to protect from the replay attack.
Since both `Home.sol` and `Replica.sol` inherit `NomadBase.sol`, the fix above will solve for both contracts.

Update:
The Nomad team decided not to take any additional steps as the EIP-712 is not chain agnostic. While this is a legitimate concern, not adding additional parameters to the signature could allow a signature to be replayed if the same contract is deployed twice with the same updater in a single chain (this could happen if multiple environments are deployed on the same chain, for example). Therefore, we recommend exploring if there's a way to add the `version`, `verifyingContract` and `salt` to the signature. Otherwise, make sure to never deploy two contracts with the same updater in a single chain.

QSP-19 Proving With An Empty Leaf

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Replica.sol`

Description: The function `Replica.sol:prove` accepts the input `_leaf` and checks if it is part of the Merkle tree. Nomad architecture uses a sparse Merkle tree, in which all the non-used leaves default with empty `bytes32`. This nature of the sparse Merkle tree makes it possible for one to pass an empty `bytes32` as the `_leaf` and some artificial Merkle proof with a specified index to pass the inclusion check. The "empty leaf" message status can later be flagged as `PROVEN`, resulting in the `messages` mapping in an undesired state.

Recommendation: Validate that the `_leaf` input of the function `Replica.sol:prove` is not empty.

Update: The Nomad team responded that "We consider it to be effectively impossible to find the preimage of the empty leaf". We believe the Nomad team has misunderstood the issue. It is not related to finding the pre-image of the empty bytes. Instead, it is about being able to prove that empty bytes are included in the tree (empty bytes are the default nodes of a sparse Merkle tree). Therefore, anyone can call the `prove` function with an empty leaf and update the status to be proven.

QSP-20 Integer Overflow /Underflow

Severity: *Low Risk*

Status: Fixed

File(s) affected: `Replica.sol`

Description: The codebase is not using `SafeMath` in the arithmetic operations. Some places can potentially be overflow or underflow in edge cases.

Recommendation: Either upgrade the solidity version to be higher than 0.8.0, where it automatically reverts when underflow and overflow, or use the `SafeMath` library to guard this.
— `Replica.sol:L153` : the `optimisticSeconds` can be set up to max of `uint256`. The calculation `block.timestamp + optimisticSeconds` is of overflow risk. For this case, we recommend defining a max value (e.g., max of uint64) for the `optimisticSeconds` in `Replica.sol:setOptimisticTimeout` and `Replica.sol:initialize`.

Update: The Nomad team added checks to ensure that the `optimisticSeconds` can never overflow. While this was the only overflow that we identified, there could be other overflow/underflow instances in the codebase or new ones could be introduced in the future. We recommend being very careful when dealing with arithmetic operations, as the contracts are still vulnerable to this kind of issues (codebase is using Solidity 0.7.6).

QSP-21 Tighten Validations Around Message

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Replica.sol`, `Message.sol`, `GovernanceMessage.sol`, `BridgeMessage.sol`

Description: Message is the first-class citizen of the Nomad system. However, it uses a dependency `TypedMemView.sol` in which the implementation aggressively uses assembly. We recommend taking extra care on all the messages and adding stricter validation to reduce the risk of an invalid message passed across the domain.

Recommendation: We recommend adding `TypedMemView.sol:assertValid` whenever `TypedMemView.sol:ref` is called to convert a byte to `TypedMemView`. The reason is that `TypedMemView.sol:ref` can return a `NULL` message if there is an error in `TypedMemView.sol:build`.
Another observation is that the implementation encodes `address` data with `bytes32`. `address` data is only 20 bytes, so there will also be 12 extra bytes in front. However, the message-related libraries do not check whether it starts with only padding 0s or not. Later, when the message is decoded in `TypeCasts.sol:bytes32ToAddress`, it ignores the first 12 bytes. Without the stricter check on address information encoding, the messages will pass dirty bytes. We recommend adding validation that those `address` encoding must start with 12 bytes of padding 0s. Following are the spotted places for the changes:

- `Message.sol:formatMessage`:
  . check that `_messageBody` should not be empty
  . `_sender` and `_recipient`: verify that the first 12 bytes can only be padding `0`.
- `Message.sol`: add a function `isValidMessage` similar to `GovernanceMessage.sol:isValidBatch` or `GovernanceMessage.sol:isValidTransferGovernor`. We can check the message length should be larger than `Message.sol:PREFIX_LENGTH`. Apply this check in `Replica.sol:L188` after it is converted to `TypedMemView` with `.ref(0)`.
- `governance/GovernanceMessage.sol:formatTransferGovernor`:
  . add validation that it starts with 12 bytes of `0`.
- `governance/GovernanceMessage.sol:serializeCall`:
  . add validation that it starts with 12 bytes of `0`.
  . add `TypedMemView.sol:assertValid` check after calling `ref(0)` in `L96`.
- `governance/GovernanceMessage.sol:getBatchHash`:
  . add `TypedMemView.sol:assertValid` check after calling `ref(0)` in `L108`.
- `BridgeMessage.sol`: add a helper function `isValidTokenId`. We should check that `domain` is not zero in the function and `id` starts with only padding `0`s and then apply this check to the following places:
  . `BridgeMessage.sol:formatMessage`: input `tokenId`

-
  . `BridgeMessage.sol:formatTokenId`: input `_domain` and `_id`.

  . `BridgeMessage.sol:getPreFillId`: input `_tokenId`

- `BridgeMessage.sol:formatTransfer`: add `TypedMemView.sol:assertValid` check after calling `ref(0)` in `L148`.
- `BridgeMessage.sol:formatTokenId`: add `TypedMemView.sol:assertValid` check after calling `ref(0)` in `L178`.
- `BridgeMessage.sol:getPreFillId`: add `TypedMemView.sol:assertValid` check after calling `ref(0)` in `L227`.
- `Replica.sol:L188`: after the line `bytes29 _m = _message.ref(0);`, we should check if the message is valid or not by calling `TypedMemView.sol:assertValid`.
- `governance/GovernanceRouter.sol:L233`: add a check with `TypedMemView.sol:assertValid` after `bytes29 _msg = _message.ref(0);`.

Update: We agree that `ref(0)` is safe after double-checking the implementation, and also `slice()` will help check the message length eventually. However, suppose we need to allow passing messages with 32 bytes of address for potential domains. In that case, we recommend at least adding the validation on `TypeCasts.sol:bytes32ToAddress` so whenever a 32 bytes data is converted to an address in Ethereum, it will not allow dirty bytes to pass.

QSP-22 Abusing Underflow

Severity: *Low Risk*

Status: Mitigated

File(s) affected: `Encoding.sol`

Description: The `Encoding.sol:L52` abuses underflow to run the for-loop. The looping variable `i` starts from 15 and decreases one by one until it gets underflow and becomes 255 to break the for-loop. The underflow mechanism will work for solidity versions before 0.8.0 as solidity 8 introduces the safe math check into the run time. Nonetheless, the `pragma` definition of the contract is `>=0.6.11`, which means it can potentially be deployed with a version higher than 0.8.0.

Recommendation: Instead of abusing underflow, run them for loop with `for (uint8 i = 15; i >= 0; i -= 1) {...}` instead.

Update: The Nomad team locked the Solidity version to 0.7.6, so the underflow will not fail (unless the file is ever updated to a newer version of Solidity). However, as a best practice, we still recommend refactoring the code as there are cleaner options available.

QSP-23 Missing Input Validation

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `Home.sol`, `Replica.sol`, `XAppConnectionManager.sol`, `BridgeRouter.sol`, `Router.sol`, `GnomadModule.sol`

Description: ```

- In `contracts-core`
  . `Home.sol:update`: there should be check on `_newRoot != _committedRoot`

  . `Home.sol:setUpdater` and `Replica.sol:setUpdater`: should check the `_updater` is not zero address. Furthermore, we can add validation that the updater should not be a contract, as the signature of the updater is required.

  . `Replica.sol:update`: there should be check on `_newRoot != _committedRoot`. Updater can try to override the `confirmAt` timestamp in `Replica.sol:L153` without this check.

  . `XAppConnectionManager.sol:unenrollReplica`:
    · `_domain` should not be 0
    · `_updater` should not be empty `bytes32.`
    · `_signature` should not be empty `bytes.`

  . `XAppConnectionManager.sol:setHome`:
    · `_home` should not be zero address

  . `XAppConnectionManager.sol:ownerEnrollReplica`:
    · `_replica` should not be zero address
    · `_domain` should not be 0
    · `_domain` should be checked with the public variable `remoteDomain` of the replica. e.g. `Replica(_replica).remoteDomain() == _domain`.

  . `XAppConnectionManager.sol:ownerUnenrollReplica`:
    · `_replica` should not be zero address

  . `XAppConnectionManager.sol:setWatcherPermission`:
    · `_watcher` should not be zero address
    · `_domain` should not be 0

  . `governance/GovernanceRouter.sol:transferRecoveryManager`: should check `_newRecoveryManager` is not zero address.

  . `governance/GovernanceRouter.sol:initialize`: should check `_xAppConnectionManager` and `_recoveryManager` is not zero address

  . `upgrade/UpgradeBeacon.sol:constructor`: should check `_controller` is a smart contract.

- In `contracts-bridge`
  . `BridgeRouter.sol:send` should check `_destination` not being 0

  . `BridgeRouter.sol:enrollCustom` should check `_domain` is not 0 and `_id` is not empty. Also, it should check that the input `_domain` is not the local domain.

  . `BridgeRouter.sol:migrate` should check `_oldRepr` is not zero address

  . `BridgeRouter.sol:preFill` should check `_origin` not being 0

- In `contracts-router`

- . `Router.sol:enrollRemoteRouter` should check `_domain` is not 0, and `_router` is not empty bytes

- In `zodiac-module-gnomad`
  . `GnomadModule.sol:constructor`
    · check `_avatar`, `_target`, `_manager` is contract. (e.g., with open-zeppelin library: doc)
    · check `_controller` is not a zero address
    · check `_controllerDomain` is not zero

  . ` GnomadModule.sol:setUp
    · check `_avatar`, `_target`, `_manager` is contract
    · check `_controller` is not a zero address
    · check `_controllerDomain` is not zero

  . `GnomadModule.sol:onlyValid`: check that `_sender` starts with only padding 0s for the first 12 bytes.

  . `GnomadModule.sol:setManager`: check `_manager` is contract

  . `GnomadModule.sol:setController`
    · check `_controller` is not a zero address
    · check `_controllerDomain` is not zero ```

Recommendation: Add validation on the input variables as put in the description.

Update: The Nomad team decided not to add additional validation as that would end up increasing gas costs.

QSP-24 Incompatibility With Deflationary Tokens

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `BridgeRouter.sol`

Description: In the `_handleTransfer` function `(L348)`, when transferring standard ERC20 deflationary tokens, the input amount may not be equal to the received amount due to the charged (and burned) transaction fee. As a result, this may not meet the assumption behind these low-level asset-transferring routines and will bring unexpected balance inconsistencies, this technique is used for the same chain.

Recommendation: Add necessary mitigation mechanisms to keep track of accurate balances. You can store the value of the balance of the contract, and after the transfer, get the new balance and calculate the difference between the old and the new value of the balance. The difference should be 0 otherwise invalid amounts will be transferred to the other side of the bridge. This technique is used only for operations in the same chain, consider using a proper design for cross chain. Also, there should be documents stating the limitation on the ERC20 tokens the system supports.

Update: The Nomad team added documentation to highlight this limitation.

QSP-25 Approve Race Condition

Severity: *Low Risk*

Status: Acknowledged

File(s) affected: `OZERC20.sol`

Description: The standard ERC20 implementation contains a widely known racing condition in it approve function, wherein a spender can witness the token owner broadcast a transaction altering their approval and quickly sign and broadcast a transaction using transferFrom to move the current approved amount from the owner's balance to the spender. If the spender's transaction is validated before the owner's, the spender will be able to get both approval amounts of both transactions.

Recommendation: Use `increaseAllowance` and `decreaseAllowance` functions to modify the approval amount instead of using the approve function to modify it.

Update: The Nomad team acknowledged the issue knowing the ubiquity of this design throughout the ecosystem, and they decided not to address this.

QSP-26 Floating Pragma

Severity: *Low Risk*

Status: Fixed

File(s) affected: `GnomadModule.sol`, `BridgeMessage.sol`

Description: The contract makes use of the floating-point pragma ^0.6.11. Contracts should be deployed using the same compiler version. Locking the pragma helps ensure that contracts are not unintentionally deployed using another pragma, such as an obsolete version, that may introduce issues in the contract system.

Recommendation: Consider locking the pragma version. It is advised that floating pragma should not be used in production. Both truffle-config.js and hardhat.config.js support locking the pragma version.

Update: The team have locked the pragma version.

QSP-27 Modifier Allows This Contract To Be The Caller

Severity: *Informational*

Status: Acknowledged

Description: The `onlyGovernor` and the `onlyGovernorOrRecoveryManager` modifiers check that the `msg.sender` is an address registered as `governor` (or `recoveryManager`) or if the address is the `GovernanceRouter` contract itself. However, the `_callLocal` method is using Solidity's low-level call function, so methods dispatched from it will have the `GovernanceRouter` as the `msg.sender`. While we couldn't find any invalid use, governance calls executed through `_callLocal` could be used to bypass the `onlyGovernor` or

`onlyGovernorOrRecoveryManager` modifiers.

Recommendation: Document that `_callLocal` is a delicate function that could bypass restrictions, so that owners are careful when dispatching governance calls. Furthermore, make sure that future refactorings do not let `_callLocal` to be called by non-privileged users.

Update: The Nomad team decided not to change the code as this behavior is by design. However, we still recommend adding documentation to the code to warn developers about the risk of bypassing restrictions by mistake when dispatching governance calls.

QSP-28 BridgeRouter Must Be The Owner Of The TokenRegistry To Work

Severity: *Informational*

Status: Mitigated

File(s) affected: `TokenRegistry.sol`, `BridgeRouter.sol`

Description: During its operation, the `BridgeRouter` contract calls the `ensureLocalToken` and `enrollCustom` on the `TokenRegistry` contract. However, these methods can only be called by the owner of the contract. In other words, the `BridgeRouter` can only work if it's the owner of the `TokenRegistry` contract.

Recommendation: Consider transferring the ownership of the `TokenRegistry` contract to the `BridgeRouter` directly from the `initialize` method to avoid any deployment misconfiguration. Also document that `BridgeRouter` must be the owner of the `TokenRegistry` for it to work properly.

Update: The Nomad team added off-chain checks to their deployment script to make sure ownership is properly transferred to the rightful owner. Thus, it will ensure that the `BridgeRouter` is the owner of the `TokenRegistry`.

QSP-29 Signature Replay Attack: Hard Fork

Severity: *Informational*

Status: Acknowledged

Description: The contracts have a hard-coded "domain" value set on deployment. The hardcoded domains will work until the chain is hard forked. Once a hard fork occurs, the signature of both chains will be considered valid on both the original and the hard-forked chain.

Exploit Scenario: 1. The team deploys Nomad on chain A and chain B.

1. One day, chain A is hard forked. Both chain A and the forked chain A' have the Nomad contracts running on top of it.
2. After the hard fork, the contract in chain A dispatches msg1, and the contract in chain A' dispatches msg2. Since msg1 and msg2 are different, now the root diverges on the forks.
3. Updater signs both the roots of msg1 and msg2.
4. Anyone can use the signature to call the function `NomadBase.sol:doubleUpdate` to prove fraud.

Recommendation: The operation team of Nomad should be aware of this and be clear on which chain to continue the service if a hard fork happens.

Update: The Nomad team stated that it is a requirement that domains are unique for disparate deployments. In the case of a hard fork, Nomad can only continue to operate on one of the chains.

QSP-30 Mass Un-enrollment

Severity: *Informational*

Status: Acknowledged

File(s) affected: `Replica.sol`, `XAppConnectionManager.sol`, `Router.sol`,

Description: From the discussion with the Nomad dev team, the default and the recommended way to add a new dapp on top of the Nomad message system is to reuse the `Replica` contract, but each `Router` will have its own `XAppConnectionManager`. The `XAppConnectionManager` can configure the watcher that the dapp trusts. Suppose the updater provides an invalid update to the `Replica`. In that case, the watcher can call `XAppConnectionManager.sol:unenrollReplica` within the `optimisticSeconds` to stop the message from being processed on the `Router` of the dapp.
However, a single invalid update on the replica will trigger un-enrollment for all the routers connected to the replica. There is a capacity (gas) limit on how many un-enrollment calls the chain can handle within the `optimisticSeconds`.

Recommendation: This should be fine with the current phase where the team plans to deploy only 2 dapps (governance and token bridge) on top of the Nomad message system. Nonetheless, suppose anyone is interested in building a dapp on top of the Nomad message system. In that case, they should be aware of this and use a new replica when a certain number of routers are already sharing the existing replica.

Update: The Nomad team believes that it's unrealistic that in the near future there will be a number of xApps so large that the blockchain will not have enough blockspace to accommodate the un-enrollment.

QSP-31 Susceptible To Signature Malleability

Severity: *Informational*

Status: Acknowledged

File(s) affected: `BridgeToken.sol`

Description: `BridgeToken.sol:permit` function receives the signature and uses `ecrecover` to get the recovered address to verify the signature. However, the `ecrecover` call does not protect from the malleable signatures of ECDSA (see: SWC-117). As a result, it will be possible for another user to replay the call with a different signature even without holding the address's private key and still passes the verification. Fortunately, the current implementation includes a `nonce` which will increase each time when building the `digest` of the signature. The immediate increase in the nonce avoids replay attacks with this vulnerability.

Recommendation: Use the open-zeppelin ECDSA library to recover. The open-zeppelin library has already included the check. Alternatively, please add the check, so we are only using lower-value `s` (`uint256(s) <= 0x7FFFFFFFFFFFFFFFFFFFFFFFFFFFFFFF5D576E7357A4501DDFE92F46681B20A0`) to mitigate the malleable signature issue.

Update: The Nomad team has acknowledged the issue that `permit` doesn't rely on signature non-malleability.

QSP-32 Missing Events

Severity: *Informational*

Status: Acknowledged

File(s) affected: `Home.sol`, `Replica.sol`, `BridgeToken.sol`, `GnomadModule.sol`

Description: Nomad architecture heavily relies on the off-chain components to sync with the on-chain status. It is crucial to emit an event on critical state updates.

Recommendation: - `Home.sol:setUpdater` and `Replica.sol:setUpdater`: `Updater` is a critical role in the whole system. If there is any update, the off-chain component should be synced immediately.

- `Replica.sol:prove`: add a `Prove` event so that the watcher or the processor can know about the newly-proven messages that the processor can process. The event is critical so the processor can see if it should call `Replica.sol:proveAndProcess` or just `Replica.sol:process`.
- `BridgeToken.sol:setDetails`: add a custom event when `_isFirstDetails` is true. Some off-chain components on the bridge operator should monitor this and update the ERC20 token information as soon as possible to avoid confusing the user who first sends the token to the domain.
- `GnomadModule.sol:setManager`: add an event when manager is updated
- `GnomadModule.sol:setController`: add an event when the controller is updated

Update: The Nomad team has added some events in their contract. However, they decided not to add additional events to the `prove` and `setDetails` methods.

QSP-33 Trusted Actors Risk And External Agents Have Too Much Power

Severity: *Informational*

Status: Acknowledged

Description: We will describe the risk from a dapp operator's point of view and a user of the dapp's perspective.
From the dapp operator's point of view, the most considerable risk is a fraudulent updater. The private key of the updater is mostly going to be in a hot wallet, as it has to sign the update frequently. If the key is ever leaked, the hacker can use the key to stop the Nomad service by signing invalid updates to the replica. The invalid update forces the dapp operator/watcher to cut off the router's replica connection. However, this usually only solves half of the problem. If we look at the standard pattern of a cross-chain action:

1. First, perform some action in the home domain.
2. Dispatch the message via the home contract
3. Updater signs the update
4. Relayer sends the update to the replica.
5. Process it in the router.

The problem is in the first step. Some action is already performed in the home domain. Although the watcher can stop invalid messages from being processed, we cannot roll back the changes on the home domain. For instance, for a token bridge, the token is already locked in the bridge router before dispatching the “transfer” message. If any fraud was to occur, we could not easily roll back the result of the token-locking in the home domain. Unless we introduce a better flow, the dapp on top of Nomad will likely need some privileged roles to recover from the failure state in the worst case.
From the dapp users' point of view, they have to trust the dapp operator (watcher). If the watcher colludes with the updater, they can process any invalid messages to the router. Also, they should be aware of the crypto-economy nature of the bond-slashing mechanism of Nomad. For instance, in a token bridge dapp, it might be best practice not to send tokens value more than X% (e.g., 50%) of the slash-able bonds from the updater so that it is less motivated to perform fraud. The correct operation of the system is completely reliant on external agents performing crucial tasks. If one or more of them stops working as expected (due to malfunction or maliciously), it could put the whole system at risk and endanger the operations happening in it. In particular, we have detected the following problems:

- Updater: Currently, there's no slashing mechanism to punish updaters that have committed fraud. Without it, external updaters will have significant incentive to participate on fraud.
- Relayer: Relayers are supposed to forward updates from the home contract to the replicas in other chains. However, relayers might delay forwarding the updates (maliciously or due to heavy load) to one or more chains. The impact from this could range from delays to full denial-of-service (messages won't be able to be delivered to their destination).
- Processor: Processors are supposed to prove and send the messages to the replica, who will eventually send them to the end recipients. However, there is no mechanism to ensure that the processor sends ALL messages or that it sends them in a timely manner; a malicious processor might decide not to send one or more messages that are not convenient to it.
- Watcher: Watchers are the agents that keep the other participants honest. They are supposed to observe homes and replicas to ensure that they are working properly. However, if watchers are not working properly (due to malfunction or if they are colluding with other malicious actors) fraudulent operations might go through the system.

Note: The documentation does not describe how external agents are chosen, rotated, how they stake their funds, and how they are slashed if a malicious operation is detected. Without this information, it's not possible to assess if the economic incentives are sufficient to discourage fraud throughout the system.

Recommendation: Design mechanisms to ensure that all external agents are always up and running and operating honestly. If agents can be external parties, make sure that staking and slashing is in place to incentivize honest behaviour. On the other hand, if agents are internal, make sure that there is enough redundancy and automated mechanisms to roll up more agents if the load is increasing or one of them is misbehaving.
Also, the team should take extra care of the private-key protection of the updater and the watcher. The Nomad team should provide documents on securing these keys in their system (e.g., using a vault or hardware security module).
The Nomad team should also guide best practices for those who would like to run a new dapp on top of the Nomad message system with documents. The document should include the security model analysis for their users.

Update: From Nomad Team "Relayers and Processors are permissionless roles with no trust assumptions. Anyone can run a Relayer or a Processor; if current actors’ operations are unsatisfactory, anyone with a vested interest (or not), can and should run one themselves. Otherwise - some component of trusted actors are explicitly part of the security model, and a generally accepted “necessary evil” for cross-domain communication (c.f. Zamyatin et al SoK on cross-chain comms https://eprint.iacr.org/2019/1128.pdf). At this stage, the lack of bonding for the Updaters is intentional, as we are working towards decentralized Updater rotation. When we move to a fully trust-minimized design, Updaters will be bonded and a rotation system will be in place. Watchers are a trusted role, but the scope of their trust is very limited. Watchers are configured by the application,not the core protocol. They have the power to halt communications for the app; they do NOT have the power to submit malicious messages (as in proof-of-authority systems like multisig or validator bridges). The best mitigation for this is to have the infrastructure and documentation in place for anyone to run their watcher (the apps themselves) or to easily provision a watcher from Nomad. A greater watcher set, in our opinion, offsets the risk inherent to the trusted design."

QSP-34 Transactions Are Marked As Processed Even If They Fail

Severity: *Undetermined*

Status: Fixed

File(s) affected: `Replica.sol`

Description: The `process` method forwards messages to their destination and mark the messages as processed regardless of the outcome of the method. If dApps do not account for this, they might end up in invalid states. For example, the `BridgeRouter` is assuming that trades sent across the chain always work and has no functionality to recover if the `handle` method fails for any reason.

Recommendation: Consider if there are cases where failed transactions (like `Out of Gas` exceptions) should not be marked as processed. Otherwise, document clearly that messages will not be processed if they fail for any reason and that it's up to the dApps to account for those failures (also determine how the `BridgeRouter` should behave if the trade fails on the receiving chain).

Update: The Nomad team refactored the code to stop marking transactions as processed when they fail. With this, now it's possible to retry messages until they succeed.

QSP-35 Upgradeable Contracts Can Be Working With Older Versions

Severity: *Undetermined*

Status: Acknowledged

Description: Communication between the different smart contracts across the system is supposed to work in perfect unison. However, if contracts are not upgraded carefully, it's possible that newer contracts will end up communicating with older (potentially not compatible) versions of other contracts; this could have unforeseen consequences.

Recommendation: Consider adding checks to the code to ensure that contracts can only communicate with compatible versions of the other contracts.

Update: The Nomad team decided to add fork tests rather than changing smart contract code.

QSP-36 No Enforcement On The Governance Messages To Be Delivered

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `GovernanceRouter.sol`

Description: On top of the regular way of sending messages, governance actions need an external agent to call the `executeCallBatch` method. If this external agent does not call this method, the batch of actions will stay forever in `Pending` status and the governance actions will never be executed.

Recommendation: Determine which external agent is in charge of executing governance actions, and design a mechanism to ensure it does so in a reliable and timely manner. Furthermore, consider documenting why governance actions have to be executed through this external agent.

Update: The Nomad team considers this behavior by design.

QSP-37 Possible To Run `executeCallBatch` While Local Governance Router Is On Recovery

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `GovernanceRouter.sol`

Description: Most methods in the `GovernanceRouter` contract are annotated with modifiers to prevent them from being executed when the contract is in recovery mode. However, the `executeCallBatch` is not annotated with any of these modifiers; it can be called regardless of the recovery status. It is unclear if this is intentional or if it's missing the right modifier. **Note:** While this method could bypass the recovery mode, the calls to execute originated from the governor router, so they are unlikely to be malicious.

Recommendation: Determine if it should be possible to call the `executeCallBatch` when the local governance router is in recovery. If it is necessary, make sure to document this decision. Otherwise, annotate the method with the `onlyNotInRecovery` modifier.

Update: The Nomad team considers this behavior by design.

QSP-38 Updater Stop Signing New Roots

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `GovernanceRouter.sol`, `Home.sol`, `BridgeRouter.sol`

Description: If the updater stops signing new messages, it can freeze the system without penalty. However, the impact on the dapps on top of it can be huge. We will analyse some exploit scenarios that this attack can cause.

- Governance Router : First, the governance router relies on the Nomad message system to dispatch the governance commands from the main governance domain to other domains. Nonetheless, the updater can attack the governance by refusing to sign the governance commands or trying to update an invalid root to force the governance router to unroll from the replica and not be able to force-drop the governance command. The risk of not applying the governance commands can be huge to the nomad system, as it is usually about system updates or incidence handling.

Fortunately, the current implementation has the role of `recoveryManager`, which can switch the `GovernanceRouter` to the recovery mode and take over the control to do actions directly on the `GovernanceRouter`. Note that there will be a one-day delay of `recoveryTimelock` according to the configuration in the deployment script.

- Updater: Secondly, the updater can also apply this attack to the token bridge. Suppose the updater denies signing the token bridge update with the current bridge implementation. In that case, it will lock the user's token without a new one minted in the destination domain. The token is locked because `BridgeRouter.sol:send` will call `IERC20(_token).safeTransferFrom` to secure the funds needed to mint on the other side of the bridge.

Exploit Scenario: loss of governor:

1. The governor calls `GovernanceRouter.sol:transferGovernor` and transfers the governor from domain 1 to domain 2.
2. The updater refused to sign the update.
3. Now, the variable `governor` in the `GovernanceRouter` in domain 1 and 2 becomes `address(0)`.

Exploit Scenario: 1. Alice sends some tokens from domain A to domain B. The token is a native ERC20 of domain A.

1. Those tokens were sent to and locked in the `BridgeRouter` contract.
2. The updater refuses to sign the signature or signs a wrong one.
3. BridgeRouter in domain B will not be called with the “transfer” message either due to no update.
4. Now, the tokens are locked, but no representation tokens are minted in the domain B.

Recommendation: A potential mitigation is to introduce slashing to the updater when not updating on time. With that, the updater must provide a signature, no matter valid or not. However, an invalid signature can be proved either by `Home.sol:improperUpdate` or `Nomad.sol:doubleUpdate`. Now, with a time limit, it patches the issue that the updater can choose not to do any update without penalty.
Note that this is a known issue by the Nomad dev team, and the solution proposed above is part of the plan to implement in the future. The current goal is to add it when they move toward the “decentralized updater” phase, where there is an actual bond for slashing and some updater rotation mechanism in place.
From our point of view, we still suggest implementing and operating with a new fraud proving function to learn more operational experience during this centralized phase. Especially, the team plans to start with “fake slashing” as in the current `UpdaterManager.sol:slashUpdater` that removes the risk of such slashing harming the team when they're still on the bootstrapping phase.

Update: The Nomad team is aware of this issue. In the future, they will decentralize this role and will introduce the necessary mechanism for selecting and rotating the Updater. Until then, the Updater is being operated by the Nomad team itself and other changes are considered out-of-scope.

QSP-39 Diverge In The Updater During Updater Rotation

Severity: *Undetermined*

Status: Acknowledged

File(s) affected: `Home.sol`, `Replica.sol`, `UpdaterManager.sol`, `NomadBase.sol`, `GovernanaceRouter.sol`

Description: To rotate the `updater`, the governor will need to send the command via the `GovernanceRouter` to call the `UpdaterManager.sol:setUpdater` function to update the `updater` in `Home` and call the `Replica.sol:setUpdater` to update the one in the `Replica`. However, the governance domain (with a valid governor) will have the updater rotated immediately, while the other domains remain the old updater until the `optimisticSeconds` (30 min) pass. During this period, the signature required in the governance domain will be different from the other domains.
Relayer relies on the `Update` event on-chain to collect the signature of the updater when `Home.sol:update` is called. The divergence of the updater across domains will cause the relayer to fail to collect the signature that the replay can send to the replica, since the home and replica can require a different signature of the updater.

Exploit Scenario: 1. Assume two domains: domain 1 and domain 2 are running with the Nomad system. Governor is on the domain 1. It is running with `updater1`.

1. `setUpdater` command is called from the governor in domain 1 to change the updater to `updater2`.
2. Since domain 1 is the primary governance domain, `GovernanceRouter.sol:executeGovernanceActions` call will change the updater immediately to `updater2`. However, domain 2 will still use `updater1` until 30 minutes pass and the message is processed.
3. Now, the `updater1` updates the root in the home contract of the domain 2.
4. When the relayer tries to send to the replica of domain 1, the signature check will fail as the current updater of domain 1 is `updater2`.

Recommendation: This can be mitigated for the initial phase if the Nomad team takes maintenance time off to stop sending new signatures from the updater for a while. In the bootstrap phase, the Nomad team will run the updater, and also, there is no plan to have actual bond slashing. It is reasonable to have off-chain coordination for the maintenance period, simply.
In the future, if a maintenance phase is acceptable, then there can be some flag on the smart contracts to freeze the `update` or `dispatch` function for a certain period whenever migrating to a new updater.
Alternatively, if zero downtime is preferred, we should change the `update` function to require two signatures during the migration phase. Both old and new updaters should sign and send the signature during the migration phase.

Update: The Nomad team state

```
The case of Fraud on the Governor domain, Fraud will be proven on the Governor Home, which will be FAILED. Thus, it will not be possible to send messages over the
normal governance channels anyway. This is the case for which the Recovery Manager role was originally designed.
* In the case of Fraud on the Governor domain, the Recovery Manager must be activated on all chains. The rotation of the Updater on Home and Replica contracts must be performed by the Recovery Manager, which is independent
of normal messaging channels.
* In the case of Fraud on other non-Governor domains, normal governance rails can be used to rotate the Updater without issue.
```

QSP-40 Initializer Not Disabled On Implementation Contract

Severity: *Informational*

Status: Acknowledged

File(s) affected: `Proxied contracts`

Description: There are several contracts in the Nomad codebase that are deployed as proxies to enable easy upgrades. When proxies are created the deployment script calls the corresponding method to initialize important functionality, like the owner of the contract. Further, calls to this method are not allowed, as it is protected by the `initializer` modifier. However, this initialize method is never called on the implementation contract itself. Therefore, an attacker could call the initializer method and take ownership of the implementation contract.

Recommendation: Consider calling the `_disableInitializers` method on the constructor of the implementation contracts to ensure that they cannot be taken over by an attacker (Check the second "Caution" callout in the OpenZeppelin Docs). We recommend doing so to prevent any unexpected behavior.

Update: The team acknowledged the issue and stated the following

```
Our upgrade structure is not controlled by the implementation. Taking control of the implementation cannot perform upgrades to malicious implementations.
We do not perform any delegateCalls in any of our implementations; there are no owner-provided values which could cause the implementation contract to self-destruct and threaten locking funds
even if the implementation contract did self-destruct, funds would not be locked.
Our upgrade structure - the UpgradeBeacon - would enable us to upgrade to a new implementation anyway.
```

Automated Analyses

Slither

Slither is run against all the contracts and found 19 issues , the majority of the issues alerted by slither are false positive.

Code Documentation

• Unclear on the recommendation of how should dapp on top of the Nomad message system set their own `Replica` and `XAppConnectionManager` contracts. From the discussion with the team, the default setup in the team's operational scenario is that different dapp will share the same `Replica`. Still, they can set their own `XAppConnectionManager` connected to their `Router` to enable their trusted watcher to unenroll the router from the replica. There should be a document on how to set up this.

• Unclear about the role of `recoveryManager` used in the `GovernanceRouter`. Also, the documents state that "If there is fraud on the Nomad Home contract on the governor chain, this is currently a "catastrophic failure state" — no further governance actions can be rolled out to remote chains; we must create a plan to recover the system in this case." in the section of failure state (link). The statement seems unrelated to the `recoveryManger`, but it might confuse the audience because the naming is the same here.

• Should warn the user that the bridge only supports standard ERC20 tokens without taking a fee on transfer or will rebase the amount.

• Lack of documents or specs on the `enrollCustom` and `migrate` features of the bridge.

• Lack of documents or specs on the fast transfer feature of the bridge.

• In package `contracts-bridge`, the code document for the function `TokenRegistry.sol:_defaultDetails` has an unfinished sentence `Sets name to "nomad.[domain].[id]" and symbol to`.

• Lack of documents on the use cases of `zodiac-module-gnomad`.

• `Home.sol`:`L57`: The param `messageHash` is not sorted in the same order as in the event. Consider sorting them in the same way.

• `Home.sol`:`L139` - The documentation is wrong. The word “it” is not needed.

• `Replica.sol`:`L67` - The documentation for the `messageHash` parameter states that it is the message that failed to process. This is incorrect.

• `Replica.sol`:`L122` - Add a comment describing why the `confirmAt` was set to 1 (probably to always pre-approve the given root).

• `XAppConnectionClient`:`L59`. typo "_potentialReplcia".

• `TokenRegistry.sol`:`L344` - The comment is incomplete and not clear. “Initialize the token separately from the ... “

• `TokenRegistry.sol`:`L364` - The comment is incomplete - “and symbol to...“

• `TokenRegistry.sol`:`L366` - Typo "pf".

• `TokenRegistry.sol`:`L155` - Typo "tje".

Adherence to Best Practices

• The `setUp` method in the `GnomadModule` contract is public even though it's only supposed to be called only once. The `__Ownable_init` call inside the method is preventing it from being called again, but this looks like an unnecessary risk (future refactorings could remove this call). If the the `setUp` method will only be called by the constructor mark it private. Otherwise, leave it as public but explicitly add the `initializer` modifier.

• `UpgradeBeaconProxy.sol` - The `_getImplementation` method in `L164` is quite confusing. It’s using Solidity’s low-level `staticcall` to call the fallback function. It would more readable and less error-prone to implement a `getImplementation()` method in the `UpgradeBeacon` contract and just call it from `UpgradeBeaconProxy.sol` as done in OpenZeppelin.

• The `UpgradeBeaconProxy` does not follow the `EIP-1967` standard. Therefore, EtherScan and other tools will not realize that this contract is dealing with a proxy. Determine if this is important and if so add support for EIP-1967 (Consider OpenZeppelin's `BeaconProxy` implementation).

• It's not clear why the constructor of the `UpgradeBeacon` and the `UpdaterManager` are marked as `payable`. If those constructors are never meant to receive `Ether` remove that keyword.

• The `Version0.sol` contract is never meant to be used on its own. Therefore, mark it as `abstract`.

• The `solhint` comment in `Replica.sol:L97` is not needed as the next code block is not empty.

• The `onlyReplica` modifier on the `XAppConnectionManager.sol` is never used (the only modifiers used are defined in `XAppConnectionClient.sol` or `GovernanceRouter.sol`). Consider removing it if not needed.

• The `ERC20` contract defined in the `OZERC20.sol` file can never be used on its own. Therefore, consider marking it as `abstract`.

• The `TokenRegistry` contract is making an assebly code on `L293` to determine if a given token is a contract. Consider replacing this call with OpenZeppelin's `Address.isContract()` or `token.code.length > 0`. Both of these options rely on `extcodesize`, but are less error prone.

• The `isLocalOrigin` method in the `TokenRegistry` contract is returning `false` if the given token address is not a contract. This is misleading, as the caller might assume that the token exists and represents a remote address. Instead, consider throwing an exception if the address is not a contract.

• The `evmId` method in the `BridgeMessage.sol` contract is never used. Please remove it if it's not necessary.

• Remove `MerkleTreeManager` and `QueueManager` from the inheritance of `Home` and remove the file `Merkle.sol` and `Queue.sol` under `contracts/`. They provided limited value by wrapping a few `tree` and `queue` functions. However, both variable `tree` and `queue` are still directly accessed in the code of `Home`. We can call those now without the wrapping calls.

• Consider merging `XAppConnectionClient` with the `Router` for simplicity.

• In `GnomadModule.sol:setUp` (`zodiac-module-gnomad`) , it should use the setter functions to set values: `Module.setAvatar`, `Module.setTarget`, `GnomadModule.setController`

• Lower down unnecessary function visibility from `internal` to `private`:

  . `contracts-core`
    - `Home.sol:_destinationAndNonce`
    - `Home.sol:_setUpdaterManager`
    - `XAppConnectionManager.sol:_recoverWatcherFromSig`
  . `contracts-bridge`
    - `BridgeRouter.sol:_originAndNonce`
    - `BridgeRouter.sol:_applyPreFillFee`
    - `BridgeRouter.sol:_dust`
    - `BridgeRouter.sol:_handleTransfer`
  . `contracts-router`
    - `XAppConnectionClient.sol:_isReplica`

-
  -
    - `Router.sol:_isRemoteRouter`
  - `zodiac-module-gnomad`
    - `GnomadModule.sol:executeTransaction`: change the visibility or just remove this function and directly call the `exec` function in the `handle` function.

- Lower down function visibility from `public` to `external`:
  - `contracts-bridge`:
    - `TokenRegistry.sol:isLocalOrigin`

- There is a magic value `0` for the "type flag" for `TypedMemView`. Instead of `<:message>.ref(0)`, please provide a readable type name instead of passing `0`.
  - `contracts-core`:
    - `Replica.sol:L188`
    - `governance/GovernanceMessage.sol:L96`
    - `governance/GovernanceMessage.sol:108`
    - `governance/GovernanceMessage.sol:159`
    - `governance/GovernanceRouter.sol:233`
    - `libs/TypeCasts.sol:L15`
  - `contracts-bridge`:
    - `BridgeMessage.sol:L148`
    - `BridgeMessage.sol:L178`
    - `BridgeMessage.sol:L227`
    - `BridgeRouter.sol:L114`
    - `BridgeRouter.sol:L217`

- Remove unused functions:
  - `contracts-core`:
    - `libs/TypeCasts.sol:coerceBytes32`
    - `libs/TypeCasts.sol:coerceString`
    - `libs/Queue.sol:dequeue(Queue storage _q, uint256 _number)`
    - `governance/GovernanceMessage.sol:typeAssert`
    - `governance/GovernanceMessage.sol:isBatch`
    - `governance/GovernanceMessage.sol:isTransferGovernor`
    - `governance/GovernanceMessage.sol:messageType`
    - `governance/GovernanceMessage.sol:mustBeBatch`
    - `governance/GovernanceRouter.sol:onlyInRecovery`
  - `contracts-bridge`
    - `BridgeMessage.sol:formatTokenId(BridgeMessage.TokenId)`
    - `BridgeRouter._localDomain`
  - `contracts-router`
    - `XAppConnectionClient.sol:_home`
    - `XAppConnectionClient.sol:_isReplica`
    - `Router.sol:_isRemoteRouter`

- Remove unused variables
  - `contracts-core`
    - `governance/GovernanceMessage.sol:BATCH_PREFIX_ITEMS`
    - `governance/GovernanceMessage.sol:CALLS_PREFIX_LEN`
    - `governance/GovernanceMessage.sol:CALL_DATA_OFFSET`
  - `contracts-bridge`
    - `BridgeMessage.sol:IDENTIFIER_LEN`

Test Results

Test Suite Results

forge test

```
YN0000: [@nomad-xyz/contracts-bridge]: Process started
 ✓ YN0000: [@nomad-xyz/contracts-core]: Process started
 ✓ YN0000: [@nomad-xyz/contracts-router]: Process started
```

```
✓ YN0000: [@nomad-xyz/multi-provider]: Process started
✓ YN0000: [@nomad-xyz/sdk]: Process started
✓ YN0000: [@nomad-xyz/multi-provider]:
✓ YN0000: [@nomad-xyz/multi-provider]:
✓ YN0000: [@nomad-xyz/multi-provider]:   multi-provider
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ returns an array of domains
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ returns array of domain numbers
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ returns an array of domain names
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ returns an array of missing providers
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ returns domain for given nameOrDomain
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ returns name for given nameOrDomain
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ resolveDomainName errors if domain is not found
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ returns whether a given domain is registered
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ fetches a domain, given a name or domain ID
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ mustGetDomain errors if a given domain is not registered
✓ YN0000: [@nomad-xyz/multi-provider]:    - registerSigner errors if no provider
✓ YN0000: [@nomad-xyz/multi-provider]:    ✓ registers provider
✓ YN0000: [@nomad-xyz/multi-provider]:    ✓ gets registered provider
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ mustGetProvider errors if provider is not registered for given nameOrDomain
✓ YN0000: [@nomad-xyz/multi-provider]:    ✓ registers signer
✓ YN0000: [@nomad-xyz/multi-provider]:    ✓ gets signers
✓ YN0000: [@nomad-xyz/multi-provider]:    ✓ gets connection
✓ YN0000: [@nomad-xyz/multi-provider]:    ✓ unregisters signer
✓ YN0000: [@nomad-xyz/multi-provider]:    - gets connection
✓ YN0000: [@nomad-xyz/multi-provider]:    - gets signer address
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ mustGetSigner errors if signer is not registered for given nameOrDomain
✓ YN0000: [@nomad-xyz/multi-provider]:    ✓ clears all signers
✓ YN0000: [@nomad-xyz/multi-provider]:    - registers Wallet Signer
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ instantiates Contracts class with appropriate args
✓ YN0000: [@nomad-xyz/multi-provider]:    - TODO: resolveDomainName errors
✓ YN0000: [@nomad-xyz/multi-provider]:
✓ YN0000: [@nomad-xyz/multi-provider]:   multi-provider utils
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ domain for given chain ID
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ gets hex domain from string
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ converts to a 32-byte cannonized ID
✓ YN0000: [@nomad-xyz/multi-provider]:     ✓ converts Nomad Id to emv address
✓ YN0000: [@nomad-xyz/multi-provider]:    - delays x milliseconds
✓ YN0000: [@nomad-xyz/multi-provider]:
✓ YN0000: [@nomad-xyz/multi-provider]:
✓ YN0000: [@nomad-xyz/multi-provider]:   24 passing (23ms)
✓ YN0000: [@nomad-xyz/multi-provider]:   6 pending
✓ YN0000: [@nomad-xyz/multi-provider]:
✓ YN0000: [@nomad-xyz/multi-provider]: Process exited (exit code 0), completed in 3s 24ms
✓ YN0000: [@nomad-xyz/sdk-bridge]: Process started
✓ YN0000: [@nomad-xyz/sdk]:
✓ YN0000: [@nomad-xyz/sdk]:
✓ YN0000: [@nomad-xyz/sdk]:   sdk
✓ YN0000: [@nomad-xyz/sdk]:     NomadContext
✓ YN0000: [@nomad-xyz/sdk]:       ✓ Is properly instantiated from a NomadConfig (54ms)
✓ YN0000: [@nomad-xyz/sdk]:       - fails if given bad rpc provider string
✓ YN0000: [@nomad-xyz/sdk]:       ✓ gets replica by name or domain
✓ YN0000: [@nomad-xyz/sdk]:       ✓ maintains connection when registering and unregistering signers
✓ YN0000: [@nomad-xyz/sdk]:     CoreContracts
✓ YN0000: [@nomad-xyz/sdk]:       - gets governor and stores in class state
✓ YN0000: [@nomad-xyz/sdk]:
✓ YN0000: [@nomad-xyz/sdk]:
✓ YN0000: [@nomad-xyz/sdk]:   3 passing (85ms)
✓ YN0000: [@nomad-xyz/sdk]:   2 pending
✓ YN0000: [@nomad-xyz/sdk]:
✓ YN0000: [@nomad-xyz/sdk]: Process exited (exit code 0), completed in 6s 776ms
✓ YN0000: [@nomad-xyz/sdk-govern]: Process started
✓ YN0000: [@nomad-xyz/sdk-bridge]:
✓ YN0000: [@nomad-xyz/sdk-bridge]:
✓ YN0000: [@nomad-xyz/sdk-bridge]:   sdk-bridge
✓ YN0000: [@nomad-xyz/sdk-bridge]:     BridgeContext
✓ YN0000: [@nomad-xyz/sdk-bridge]:       ✓ Is properly instantiated from a NomadConfig and then NomadContext
✓ YN0000: [@nomad-xyz/sdk-bridge]:     Nomad events
✓ YN0000: [@nomad-xyz/sdk-bridge]:       - sends bridge transaction
✓ YN0000: [@nomad-xyz/sdk-bridge]:
✓ YN0000: [@nomad-xyz/sdk-bridge]:
✓ YN0000: [@nomad-xyz/sdk-bridge]:   1 passing (79ms)
✓ YN0000: [@nomad-xyz/sdk-bridge]:   1 pending
✓ YN0000: [@nomad-xyz/sdk-bridge]:
✓ YN0000: [@nomad-xyz/sdk-bridge]: Process exited (exit code 0), completed in 6s 671ms
✓ YN0000: [@nomad-xyz/contracts-core]: Compiling 47 files with 0.7.6
✓ YN0000: [@nomad-xyz/contracts-core]: Solc 0.7.6 finished in 4.80s
✓ YN0000: [@nomad-xyz/contracts-core]: Compiler run successful (with warnings)
✓ YN0000: [@nomad-xyz/contracts-core]: warning[2462]: node_modules/@openzeppelin/contracts/access/Ownable.sol:26:5: Warning: Visibility for constructor is ignored. If you want the contract to be non-deployable, making it
"abstract" is sufficient.
✓ YN0000: [@nomad-xyz/contracts-core]:     constructor () internal {
✓ YN0000: [@nomad-xyz/contracts-core]:     ^ (Relevant source part starts here and spans across multiple lines).
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: warning[2072]: packages/contracts-core/contracts/test/Home.t.sol:83:9: Warning: Unused local variable.
✓ YN0000: [@nomad-xyz/contracts-core]:         bytes memory message = Message.formatMessage(
✓ YN0000: [@nomad-xyz/contracts-core]:         ^------------------^
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: warning[2072]: packages/contracts-core/contracts/test/utils/MerkleTest.sol:22:9: Warning: Unused local variable.
✓ YN0000: [@nomad-xyz/contracts-core]:         bytes32 hash = keccak256(message);
✓ YN0000: [@nomad-xyz/contracts-core]:         ^----------^
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: warning[2072]: packages/contracts-core/contracts/test/Replica.t.sol:232:13: Warning: Unused local variable.
✓ YN0000: [@nomad-xyz/contracts-core]:             bytes32 leaf,
✓ YN0000: [@nomad-xyz/contracts-core]:             ^----------^
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: warning[2072]: packages/contracts-core/contracts/test/Replica.t.sol:533:13: Warning: Unused local variable.
✓ YN0000: [@nomad-xyz/contracts-core]:             bytes32 leaf,
✓ YN0000: [@nomad-xyz/contracts-core]:             ^----------^
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: warning[2072]: packages/contracts-core/contracts/test/Replica.t.sol:534:13: Warning: Unused local variable.
✓ YN0000: [@nomad-xyz/contracts-core]:             uint256 index,
✓ YN0000: [@nomad-xyz/contracts-core]:             ^-----------^
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: warning[2072]: packages/contracts-core/contracts/test/Replica.t.sol:535:13: Warning: Unused local variable.
✓ YN0000: [@nomad-xyz/contracts-core]:             bytes32[32] memory proof
✓ YN0000: [@nomad-xyz/contracts-core]:             ^----------------------^
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: Running 6 tests for packages/contracts-core/contracts/test/NomadBase.t.sol:NomadBaseTest
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptUpdaterSignature() (gas: 27469)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_failInitializeTwice() (gas: 13452)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_homeDomainHash() (gas: 10078)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_ownerIsContractCreator() (gas: 7631)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_rejectNonUpdaterSignature() (gas: 27434)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_stateIsActiveAfterInit() (gas: 7679)
✓ YN0000: [@nomad-xyz/contracts-core]: Test result: ok. 6 passed; 0 failed; finished in 1.12s
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: Running 8 tests for packages/contracts-core/contracts/test/Home.t.sol:HomeTest
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_committedRoot() (gas: 13674)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_dispatchRejectBigMessage() (gas: 29181)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_homeDomain() (gas: 8010)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_improperUpdate() (gas: 46103)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_nonUpdaterManagerCannotSetUpdater() (gas: 11839)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_onlyUpdaterManagerSetUpdater() (gas: 19666)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_succesfulDispatch() (gas: 216412)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_successfulDispatchAndUpdate() (gas: 228379)
✓ YN0000: [@nomad-xyz/contracts-core]: Test result: ok. 8 passed; 0 failed; finished in 1.13s
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: Running 24 tests for packages/contracts-core/contracts/test/Replica.t.sol:ReplicaTest
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptLeafCorrectProof() (gas: 141654)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptReplicaUpdate() (gas: 66564)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptableRootLegacyRejectStatus() (gas: 10263)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptableRootLegacySuccess() (gas: 6361)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptableRootRejectNotCommited() (gas: 7941)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptableRootRejectNotTimedOut() (gas: 96705)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptableRootSuccess() (gas: 9985)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyProvenMessageEmptyAddress() (gas: 46553)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyProvenMessageRevertingHandlers1() (gas: 48959)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyProvenMessageRevertingHandlers2() (gas: 49086)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyProvenMessageRevertingHandlers3() (gas: 49018)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyProvenMessageRevertingHandlers4() (gas: 48958)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyWrongDestination() (gas: 68756)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessUnprovenMessage() (gas: 48486)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_processLegacyProvenMessageReturnZeroHandler() (gas: 55453)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_processProvenMessage() (gas: 127782)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_proveAndProcess() (gas: 130139)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_rejectLeafWrongProof() (gas: 126638)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_rejectReplicaNonCurrentUpdate() (gas: 17008)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_rejectReplicaUpdateInvalidSig() (gas: 27613)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_setConfirmationOnlyOwner() (gas: 39727)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_setOptimisticTimeoutOnlyOwner() (gas: 18541)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_setUpdaterOnlyOwner() (gas: 25159)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_updateProveAndProcessMessage() (gas: 149999)
✓ YN0000: [@nomad-xyz/contracts-core]: Test result: ok. 24 passed; 0 failed; finished in 1.13s
✓ YN0000: [@nomad-xyz/contracts-core]: No files changed, compilation skipped
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: Running 8 tests for packages/contracts-core/contracts/test/Home.t.sol:HomeTest
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_committedRoot() (gas: 13674)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_dispatchRejectBigMessage() (gas: 29181)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_homeDomain() (gas: 8010)
```

```
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_improperUpdate() (gas: 46103)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_nonUpdaterManagerCannotSetUpdater() (gas: 11839)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_onlyUpdaterManagerSetUpdater() (gas: 19666)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_succesfulDispatch() (gas: 216412)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_successfulDispatchAndUpdate() (gas: 228379)
✓ YN0000: [@nomad-xyz/contracts-core]: Test result: ok. 8 passed; 0 failed; finished in 822.77ms
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: Running 24 tests for packages/contracts-core/contracts/test/Replica.t.sol:ReplicaTest
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptLeafCorrectProof() (gas: 141654)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptReplicaUpdate() (gas: 66564)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptableRootLegacyRejectStatus() (gas: 10263)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptableRootLegacySuccess() (gas: 6361)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptableRootRejectNotCommited() (gas: 7941)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptableRootRejectNotTimedOut() (gas: 96705)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptableRootSuccess() (gas: 9985)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyProvenMessageEmptyAddress() (gas: 46553)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyProvenMessageRevertingHandlers1() (gas: 48959)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyProvenMessageRevertingHandlers2() (gas: 49086)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyProvenMessageRevertingHandlers3() (gas: 49018)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyProvenMessageRevertingHandlers4() (gas: 48958)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessLegacyWrongDestination() (gas: 68756)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_notProcessUnprovenMessage() (gas: 48486)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_processLegacyProvenMessageReturnZeroHandler() (gas: 55453)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_processProvenMessage() (gas: 127782)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_proveAndProcess() (gas: 130139)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_rejectLeafWrongProof() (gas: 126638)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_rejectReplicaNonCurrentUpdate() (gas: 17008)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_rejectReplicaUpdateInvalidSig() (gas: 27613)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_setConfirmationOnlyOwner() (gas: 39727)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_setOptimisticTimeoutOnlyOwner() (gas: 18541)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_setUpdaterOnlyOwner() (gas: 25159)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_updateProveAndProcessMessage() (gas: 149999)
✓ YN0000: [@nomad-xyz/contracts-core]: Test result: ok. 24 passed; 0 failed; finished in 822.82ms
✓ YN0000: [@nomad-xyz/contracts-core]:
✓ YN0000: [@nomad-xyz/contracts-core]: Running 6 tests for packages/contracts-core/contracts/test/NomadBase.t.sol:NomadBaseTest
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_acceptUpdaterSignature() (gas: 27469)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_failInitializeTwice() (gas: 13452)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_homeDomainHash() (gas: 10078)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_ownerIsContractCreator() (gas: 7631)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_rejectNonUpdaterSignature() (gas: 27434)
✓ YN0000: [@nomad-xyz/contracts-core]: [PASS] test_stateIsActiveAfterInit() (gas: 7679)
✓ YN0000: [@nomad-xyz/contracts-core]: Test result: ok. 6 passed; 0 failed; finished in 829.94ms
✓ YN0000: [@nomad-xyz/contracts-core]: Process exited (exit code 0), completed in 10s 720ms
✓ YN0000: [@nomad-xyz/contracts-router]: Generating typings for: 25 artifacts in dir: ./src for target: ethers-v5
✓ YN0000: [@nomad-xyz/contracts-router]: Successfully generated 31 typings!
✓ YN0000: [@nomad-xyz/contracts-router]: Compiled 25 Solidity files successfully
✓ YN0000: [@nomad-xyz/contracts-router]: ·-----------------------|--------------------------|----------------|-----------------------------·
✓ YN0000: [@nomad-xyz/contracts-router]: |  Solc version: 0.7.6  ·  Optimizer enabled: true  ·  Runs: 999999  ·  Block limit: 30000000 gas  |
✓ YN0000: [@nomad-xyz/contracts-router]: ··························|··························|·················|·····························
✓ YN0000: [@nomad-xyz/contracts-router]: |  Methods                                                                                        |
✓ YN0000: [@nomad-xyz/contracts-router]: ·············|··········|·············|·············|·················|·············|··············
✓ YN0000: [@nomad-xyz/contracts-router]: |  Contract  ·  Method  ·  Min        ·  Max        ·  Avg            ·  # calls    ·  usd (avg)    |
✓ YN0000: [@nomad-xyz/contracts-router]: ·------------|----------|-------------|-------------|----------------|-------------|---------------·
✓ YN0000: [@nomad-xyz/contracts-router]:
✓ YN0000: [@nomad-xyz/contracts-router]:   0 passing (89ms)
✓ YN0000: [@nomad-xyz/contracts-router]:
✓ YN0000: [@nomad-xyz/contracts-router]: @openzeppelin/contracts/access/Ownable.sol:26:5: Warning: Visibility for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient.
✓ YN0000: [@nomad-xyz/contracts-router]:     constructor () internal {
✓ YN0000: [@nomad-xyz/contracts-router]:     ^ (Relevant source part starts here and spans across multiple lines).
✓ YN0000: [@nomad-xyz/contracts-router]:
✓ YN0000: [@nomad-xyz/contracts-router]: Process exited (exit code 0), completed in 10s 735ms
✓ YN0000: [@nomad-xyz/sdk-govern]:
✓ YN0000: [@nomad-xyz/sdk-govern]:
✓ YN0000: [@nomad-xyz/sdk-govern]:   sdk-govern
✓ YN0000: [@nomad-xyz/sdk-govern]:     - TODO
✓ YN0000: [@nomad-xyz/sdk-govern]:
✓ YN0000: [@nomad-xyz/sdk-govern]:
✓ YN0000: [@nomad-xyz/sdk-govern]:   0 passing (1ms)
✓ YN0000: [@nomad-xyz/sdk-govern]:   1 pending
✓ YN0000: [@nomad-xyz/sdk-govern]:
✓ YN0000: [@nomad-xyz/sdk-govern]: Process exited (exit code 0), completed in 4s 339ms
✓ YN0000: [@nomad-xyz/contracts-bridge]: Generating typings for: 45 artifacts in dir: ./src for target: ethers-v5
✓ YN0000: [@nomad-xyz/contracts-bridge]: Successfully generated 63 typings!
✓ YN0000: [@nomad-xyz/contracts-bridge]: Compiled 44 Solidity files successfully
✓ YN0000: [@nomad-xyz/contracts-bridge]: ·-----------------------|--------------------------|----------------|-----------------------------·
✓ YN0000: [@nomad-xyz/contracts-bridge]: |  Solc version: 0.7.6  ·  Optimizer enabled: true  ·  Runs: 999999  ·  Block limit: 30000000 gas  |
✓ YN0000: [@nomad-xyz/contracts-bridge]: ··························|··························|·················|·····························
✓ YN0000: [@nomad-xyz/contracts-bridge]: |  Methods                                                                                        |
✓ YN0000: [@nomad-xyz/contracts-bridge]: ·············|··········|·············|·············|·················|·············|··············
✓ YN0000: [@nomad-xyz/contracts-bridge]: |  Contract  ·  Method  ·  Min        ·  Max        ·  Avg            ·  # calls    ·  usd (avg)    |
✓ YN0000: [@nomad-xyz/contracts-bridge]: ·------------|----------|-------------|-------------|----------------|-------------|---------------·
✓ YN0000: [@nomad-xyz/contracts-bridge]:
✓ YN0000: [@nomad-xyz/contracts-bridge]:   0 passing (169ms)
✓ YN0000: [@nomad-xyz/contracts-bridge]:
✓ YN0000: [@nomad-xyz/contracts-bridge]: @openzeppelin/contracts/access/Ownable.sol:26:5: Warning: Visibility for constructor is ignored. If you want the contract to be non-deployable, making it "abstract" is sufficient.
✓ YN0000: [@nomad-xyz/contracts-bridge]:     constructor () internal {
✓ YN0000: [@nomad-xyz/contracts-bridge]:     ^ (Relevant source part starts here and spans across multiple lines).
✓ YN0000: [@nomad-xyz/contracts-bridge]:
✓ YN0000: [@nomad-xyz/contracts-bridge]: contracts/BridgeRouter.sol:324:13: Warning: Failure condition of 'send' ignored. Consider using 'transfer' instead.
✓ YN0000: [@nomad-xyz/contracts-bridge]:             payable(_recipient).send(DUST_AMOUNT);
✓ YN0000: [@nomad-xyz/contracts-bridge]:             ^-----------------------------------^
✓ YN0000: [@nomad-xyz/contracts-bridge]:
✓ YN0000: [@nomad-xyz/contracts-bridge]: Process exited (exit code 0), completed in 11s 925ms
✓ YN0000: Done in 11s 927ms
```

Appendix

File Signatures

The following are the SHA-256 hashes of the reviewed files. A file with a different SHA-256 hash has been modified, intentionally or otherwise, after the security review. You are cautioned that a different SHA-256 hash could be (but is not necessarily) an indication of a changed condition or potential vulnerability that was not within the scope of the review.

Contracts

20e407301c6bf1c41e6f706fee763c3ec58617338a4ca2df6f3cae22a37ea5a9 ./Nomad/zodiac/contracts/GnomadModule.sol

8f98be62dc7d0ac20685740d920a9c78753888b39c658de7eacaa202b61dd2f0 ./Nomad/zodiac/contracts/test/Imports.sol

3bf9aa81de72cd8e4522b5ed546bab4dd63f3a7b7502e62489d2f44f021e3aab ./Nomad/zodiac/contracts/test/Mock.sol

597af176ac262ad9e23eaf0d2306d275d50640ecc7aa8fe1c381524de04b1638 ./Nomad/zodiac/contracts/test/MockConnectionManager.sol

9040a0d30df69a7a266e2e1706039aa518b6ec124f697390b8a3edf50a2fdef4 ./Nomad/zodiac/contracts/test/TestAvatar.sol

203abf1a6bd49f0285ee9e3df54b22f7c51ccee3eb271a0297a7acb3e79a10df ./Nomad/zodiac/contracts/test/TestFactory.sol

eb0167b1c14cef3031e76e798268da52fd19d43c30331f502f95bc5d5ad252f3 ./Nomad/monrepo/packages/local-environment/tests/utils/token/contract/Context.sol

60c3bb830c82a94e21fc9e4ece1400282ba46d1244afd251404735cbb6e72825 ./Nomad/monrepo/packages/local-environment/tests/utils/token/contract/ERC20.sol

6a7b7a3f5b0f0011b42bb3ae925ada6b06a24633c63b38d8bce6234a7313f034 ./Nomad/monrepo/packages/local-environment/tests/utils/token/contract/IERC20.sol

8e51e13ae9e0df1dd2336d8f7a20620bc3551e1f5719af4eabe48938684c127a ./Nomad/monrepo/packages/local-environment/tests/utils/token/contract/Token.sol

b6b3e1a64bb480a8315a55946678c8e290412f5e3ed817f8bae9e819d86f8960 ./Nomad/monrepo/packages/contracts-router/contracts/Router.sol

018f5299b11762b2d2bdb3274fe11f6ec9dbfa8295f65fce5e7d97b9350f8f18 ./Nomad/monrepo/packages/contracts-router/contracts/XAppConnectionClient.sol

d00a117e25bc8ba1071845d331028a3c58cb5dd5d1e42a2f8571c425aef2196d ./Nomad/monrepo/packages/contracts-core/contracts/Home.sol

d0f40e08a0ed29e6fb3e208a412a17f2c07db17a558dac7091c61686088ed5c3 ./Nomad/monrepo/packages/contracts-core/contracts/Merkle.sol

de15ff842258e7e22af1bfd2183a86f3d183bd6f416040c290e7c2f4966718a9 ./Nomad/monrepo/packages/contracts-core/contracts/NomadBase.sol

7fb0fd531a9f2283ef1cbd3c24e2d0fa3feb8b1ddc928e19f53e4544fe031a41 ./Nomad/monrepo/packages/contracts-core/contracts/Queue.sol

```
4b9de559b63b90a12907be0931e8769ee406642312574e52d3349a71355e1831 ./Nomad/monrepo/packages/contracts-core/contracts/Replica.sol

d1ad5ebafc193db8f0723b9e38b0c9c35a89c3739cab44c6fa88fd2247b0477f ./Nomad/monrepo/packages/contracts-core/contracts/UpdaterManager.sol

e2b284e9446b30386024b9aa0e1500488a3fc7d0e8aa747a45aa4fe6590263a3 ./Nomad/monrepo/packages/contracts-core/contracts/Version0.sol

f6ef802dd25c3abed7fe87c00c7692e286cba4b608ad2eadb4dd65733ee81bb1 ./Nomad/monrepo/packages/contracts-core/contracts/XAppConnectionManager.sol

1c0e101260c2549940880292443e755989cf8c61b871bc66b6626b247503965d ./Nomad/monrepo/packages/contracts-core/contracts/upgrade/UpgradeBeacon.sol

832ffa316e24d689c19d7023748a267712d84d3cf6a7e3f90289be808cebad15 ./Nomad/monrepo/packages/contracts-
core/contracts/upgrade/UpgradeBeaconController.sol

6bf68a5b556dac76487fb922a5c5ba1a373e67f57368dce683e8ce07619d79d8 ./Nomad/monrepo/packages/contracts-
core/contracts/upgrade/UpgradeBeaconProxy.sol

1d8f79d86f912ed20c44266a7296efb01ebe466af521d2bbd4b3beb990567dd3 ./Nomad/monrepo/packages/contracts-core/contracts/test/MysteryMath.sol

4ea99fa2bf1e686dd296d71e5c358c7b306ec926b9fcfcf2b50214058a242adf ./Nomad/monrepo/packages/contracts-core/contracts/test/MysteryMathV1.sol

47adc30c71b54462030de2e089c102ac27fb0de682f4921bd8f6559499abff38 ./Nomad/monrepo/packages/contracts-core/contracts/test/MysteryMathV2.sol

7f6beb01d945d5d32901f21ad8f52ba8dbe932e10d68e6580a2e3e7eac8e4503 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestCommon.sol

f30ca27e325e08cfec07a382dc2cb41512c4d1832906c4d390ae59e5911e969f ./Nomad/monrepo/packages/contracts-
core/contracts/test/TestGovernanceMessage.sol

2a93462dde81dc77165785e9c7a780e14ba44d93ebbcb0ae9ec545eb507ddb6e ./Nomad/monrepo/packages/contracts-
core/contracts/test/TestGovernanceRouter.sol

f2ee9ecb9a7717f6c59283b076fe779450092118015915e4edc6e7935b7095f3 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestHome.sol

668a6c81374b08925970d8ed2c09b15f044c56c51b45e33a7a2d98f5c93605f0 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestMerkle.sol

0cea4931ae64899e37f4282343555240ebd51a77c24dd6942903d75413219276 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestMessage.sol

8192727c23fb18126efbe62c3901164ac468ff8d98b3cabdfedb74ec7345540e ./Nomad/monrepo/packages/contracts-core/contracts/test/TestQueue.sol

63f5b9ef69d92b9b8fd50d08040ea95cc6fcaee3dc1f21a38b4fdfbc80404b33 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestRecipient.sol

87c75a1dfd272782de15bd380bc650c6fc05aef8f5621f770eee2ede492a7bb0 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestReplica.sol

7fe0ca4b7ac0666f1625ef65159a338f2943cabe7b6668e7f4844d30081eee04 ./Nomad/monrepo/packages/contracts-
core/contracts/test/TestXAppConnectionManager.sol

32dcd82fa0e32e3745eca4f9e39458eee1fb0683d22d3dc6b6b99638a3aabcb2 ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient1.sol

df57b1d5fc8f518c0d13ad72894445b8bf4efc45c13c540289aed9055d911a99 ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient2.sol

a252945415af04654838bdc3959c979219965eb31ab88016cf7c624a5e4b0e4c ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient3.sol

3ba8e0c4f42fa62321b94b58756488851b5c365ed4722ca306d4bc265b1385c4 ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient4.sol

e9d43d25bc487b1b8a5335adeec84197bcf16d0910be6f30740333bd258a98f4 ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient5.sol

2ebbec7b7fc89f2d49dc473cbedf6a7f1f54abd359fc3f78317062497b17e636 ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient6.sol

63b9010a994b096ea2d787b38921b0f995543b15f19f07a6c33e5c1a90eafeaf ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipientHandle.sol

ba936cbe96f54b719623b6c95a2f069356fe14e8061196d9f02884ddf1deffc8 ./Nomad/monrepo/packages/contracts-core/contracts/libs/Merkle.sol

e2262f0517d1d69c5d200b3ac6998134ca24ff4737b89e0514955459f650c67 ./Nomad/monrepo/packages/contracts-core/contracts/libs/Message.sol

25bf0b13e085ab095235ff25bf842b5b2b03d0992afc4aec2cec8d3ae2403b60 ./Nomad/monrepo/packages/contracts-core/contracts/libs/Queue.sol

b82274ebdf155c9cd1750a64b864a54c17c79cf59188b7c58e33f3fe2c09e21c ./Nomad/monrepo/packages/contracts-core/contracts/libs/TypeCasts.sol

40479a0d3596ddcd00fa2ff6e24cee120617e784bca4ec4413e461e383ae89e4 ./Nomad/monrepo/packages/contracts-
core/contracts/interfaces/IMessageRecipient.sol

7958f62ba4adb2d6367d94a8736585189e7b70b7ff5794fc22770a267c35b9a1 ./Nomad/monrepo/packages/contracts-
core/contracts/interfaces/IUpdaterManager.sol

a165f6a668b15714bccb4b495372ddf87d68cc5d21f8481cb27475cfddca4620 ./Nomad/monrepo/packages/contracts-
core/contracts/governance/GovernanceMessage.sol

7d80a2fd57a7dd4bf28b858b8552c355a32629dcb8a156ddcee8b7f2b4bec2c8 ./Nomad/monrepo/packages/contracts-
core/contracts/governance/GovernanceRouter.sol

a06c0a999cf1d52ef4afe30a315241c33c3d711c164897e1c94fd78e86d7502c ./Nomad/monrepo/packages/contracts-bridge/contracts/BridgeMessage.sol

70bfd5e9872f2d5c69e97d9bc00c487460a1643000888aeef8ae5ecd5a49ee59 ./Nomad/monrepo/packages/contracts-bridge/contracts/BridgeRouter.sol

471e85f1fb02b5fb766bee82ff222cffd6d1c520e4bf477b07a653633a43122a ./Nomad/monrepo/packages/contracts-bridge/contracts/BridgeToken.sol

8e5faf5a8fb075aec05853c8c40e1cc258de4adeb4b85f9edd646a6bed1c5c45 ./Nomad/monrepo/packages/contracts-bridge/contracts/Encoding.sol

fd1783d494ee70cff5b0015e6b743792b35d92d4516e23199a07f4a7545a3cbd ./Nomad/monrepo/packages/contracts-bridge/contracts/ETHHelper.sol

a730af6c84a6774f449fd2516b0be0c40cb5efd43f33d803c8c5530bc54d82a1 ./Nomad/monrepo/packages/contracts-bridge/contracts/TokenRegistry.sol

c9f0d8defff61d9f6268b0d198f9aa64c687d1659dc38b134c46c9ee5e43a93 ./Nomad/monrepo/packages/contracts-bridge/contracts/vendored/OZERC20.sol

cba19a6b53ed08db707391b2998e8cdcb7141357738d37a677e51bf07e9eba4d ./Nomad/monrepo/packages/contracts-bridge/contracts/test/MockCore.sol

eae58a9770879c46b0fac491d837e2def75f6646faa98e93639b2d7a9ab65dec ./Nomad/monrepo/packages/contracts-bridge/contracts/test/MockWeth.sol

dcf6ab9f802fa128422be1a18798478ae81f2f552a17888d49a596f2a3aeb1e9 ./Nomad/monrepo/packages/contracts-
bridge/contracts/test/TestBridgeMessage.sol

abb23624cc9441d042cd697d263a1cd96ef5109a9ea05deb2e8366c5bb8448e7 ./Nomad/monrepo/packages/contracts-bridge/contracts/test/TestBridgeRouter.sol

ea8d6853a57b3b86065bdf7a59dd271922d9c25e296c5cac00e7b75ef0414e0b ./Nomad/monrepo/packages/contracts-bridge/contracts/test/TestEncoding.sol

e70586fb593fd2da4827c1e4d0171c213d83af2f87761fdf53dbc51333860e80 ./Nomad/monrepo/packages/contracts-
bridge/contracts/interfaces/IBridgeToken.sol

28ff236d43f9be5f244554759ce1f710a6ff8c69916b74d20f400e5816a3fbb6 ./Nomad/monrepo/packages/contracts-
bridge/contracts/interfaces/ITokenRegistry.sol

6601834c30605d07982020a1f0788824500f55278c528ead3721f97a429dffd5 ./Nomad/monrepo/packages/contracts-bridge/contracts/interfaces/IWeth.sol
```

Tests

f843d4ead29f176148bb677bb53adb998dd452c106163fc2818904fcf840b567 ./Nomad/zodiac/.eslintrc.js

de2529a8e1cf5b57ecaf2b2722e202306b99d332a96d9d619d3a0d380ef32bd1 ./Nomad/zodiac/.solcover.js

fda9bc2d1013fefc3e86517667da943b827d110b41fb55c08959772fe0b65c5a ./Nomad/zodiac/hardhat.config.ts

40215710992cca1fdc17f66e11e8b5db878781521395f778214720ad9d438a9a ./Nomad/zodiac/test/FactoryFriendly.spec.ts

af279237e61e95993a352f05b652cd7d70962a11cf6789294bfcfb53e1c43b7d ./Nomad/zodiac/test/GnomadModule.spec.ts

4240c433d6603544d1564a46a6a29b4bf07d149b1d3168d6cbc36f1b497561c6 ./Nomad/zodiac/test/utils.ts

fbcefebfe2e23706743b13d7d277ce40a2dc43883b28158469023a6ce00c2280 ./Nomad/zodiac/src/deploy/deploy_module.ts

ff6659aa31171f904bb37db7dbc20fb2289761ece81e30a6251758f1a9d26f03 ./Nomad/zodiac/src/deploy/verify.ts

20e407301c6bf1c41e6f706fee763c3ec58617338a4ca2df6f3cae22a37ea5a9 ./Nomad/zodiac/contracts/GnomadModule.sol

8f98be62dc7d0ac20685740d920a9c78753888b39c658de7eacaa202b61dd2f0 ./Nomad/zodiac/contracts/test/Imports.sol

3bf9aa81de72cd8e4522b5ed546bab4dd63f3a7b7502e62489d2f44f021e3aab ./Nomad/zodiac/contracts/test/Mock.sol

597af176ac262ad9e23eaf0d2306d275d50640ecc7aa8fe1c381524de04b1638 ./Nomad/zodiac/contracts/test/MockConnectionManager.sol

9040a0d30df69a7a266e2e1706039aa518b6ec124f697390b8a3edf50a2fdef4 ./Nomad/zodiac/contracts/test/TestAvatar.sol

203abf1a6bd49f0285ee9e3df54b22f7c51ccee3eb271a0297a7acb3e79a10df ./Nomad/zodiac/contracts/test/TestFactory.sol

2fb45566320ea8b2e5743c8f40d8b907e25777f12463a24687a99d9aa7d60a74 ./Nomad/monrepo/packages/sdk-govern/tests/index.test.ts

164e766c402f9f073900d39e89de8d24e5278bb4ebbd0cdc59c3d6f102633c6d ./Nomad/monrepo/packages/sdk-govern/src/governanceEvents.ts

2b5c133ab81fc88906b21e38c974d025e28e9fb416581dc558e3b9443e9fffb6 ./Nomad/monrepo/packages/sdk-govern/src/GovernanceMessage.ts

ee3a4c16f22e541286f267fe2cbb26b75c1450db6ce35a51fe26f2fbb9d645f1 ./Nomad/monrepo/packages/sdk-govern/src/index.ts

ce0427d12824ff043e31c6d8d7ed069e93ecc618014864b8f1e7040af6c3d57c ./Nomad/monrepo/packages/sdk-govern/src/utils.ts

68474b47469735a7664619376a2a073a1b7b41b6b1524992a9d619408021e4e5 ./Nomad/monrepo/packages/sdk-bridge/tests/index.test.ts

a9d54cec9411a51dc8162dd3a1b5204c10e8c5a626efce5eeb41dea03b5daa49 ./Nomad/monrepo/packages/sdk-bridge/src/BridgeContext.ts

dfd4e1f20cd871fcf9c048b288543f4b4a3f867331b927a1f01cd6b6cff8f4b4 ./Nomad/monrepo/packages/sdk-bridge/src/BridgeContracts.ts

7887b2e91070642e9381fb581d54782104de9fe6a65a2adf243937b28282cd63 ./Nomad/monrepo/packages/sdk-bridge/src/bridgeEvents.ts

47c798622b433f789f07d2470f230e984e645a0495ebc68183a99998e3bfd96b ./Nomad/monrepo/packages/sdk-bridge/src/BridgeMessage.ts

00adb4a238b66f5ddd1b797e4bd51bb59c612ae7f97896aeeecbf628ba8914ac ./Nomad/monrepo/packages/sdk-bridge/src/index.ts

ac02c2238f05533b4bf34937d2151bdd5e7fec1c9b329d8ec278e76e0c27345e ./Nomad/monrepo/packages/sdk-bridge/src/sendCoins.ts

e419bcf785e5a32304c3e943e608c552a92e5115c5b9e2313cfa8bffeece3908 ./Nomad/monrepo/packages/sdk-bridge/src/tokens/index.ts

7476f4f8d6e7e2b7437d21acc35212eb8aa0a0ac221f1fc55c41b17e0dbd931d ./Nomad/monrepo/packages/sdk-bridge/src/tokens/testnetWellKnown.ts

6696446d6ccb165c36c14f008c804a4d40fdbf1de9d13103d1a6d8fcc8a478ed ./Nomad/monrepo/packages/sdk-bridge/src/tokens/wellKnown.ts

b2ad274f4379fd6c2cd7497796d3901451117b2cb0d285d8b4cffdd9aa4be20a ./Nomad/monrepo/packages/sdk/tests/index.test.ts

ea5d8cab505ce022deb1c2dceed419c097dee81483d1937b4fbd2ce55a226b51 ./Nomad/monrepo/packages/sdk/src/CoreContracts.ts

f6f91bf59044c7a50a9fb4127c70612aaee79da06d01035d17f7e59059c7b6fb ./Nomad/monrepo/packages/sdk/src/error.ts

d96f01448847f99fa0a5e8a41093467d29a2863e8d0bd4f9b0636d431d0e4b86 ./Nomad/monrepo/packages/sdk/src/index.ts

9366793a45d2e978b0428947b1d2b751d23d53811aaba49b9b027081c6be4e1a ./Nomad/monrepo/packages/sdk/src/NomadContext.ts

acc7b067fb52b02bf6f120bc946ad6fbbaae25248434ddba9ad4f3268ecd0e50 ./Nomad/monrepo/packages/sdk/src/messages/index.ts

188678b7f55b06bbad12fbd9678ab2e8fd65b8518c2cdd853ead2d5dc8627ebd ./Nomad/monrepo/packages/sdk/src/messages/NomadMessage.ts

d9e495053858c4d335ac2872e49155f7c625ecf8a1211e9fbb0e42265f8d8c1c ./Nomad/monrepo/packages/sdk/src/events/fetch.ts

cdacce67bfb8b491de65fa9d5aa6d1ec3df0bd3a9e91df4b1624ee9783ac9d2b ./Nomad/monrepo/packages/sdk/src/events/index.ts

7e2d58c6016186fc3041456e77b916d10c1418db4521f9c7de745772671fe49c ./Nomad/monrepo/packages/sdk/src/events/nomadEvents.ts

d671d2aa72efd946593fba0397fccb8ff1eac98ac71a662a55b80f6ae64d5813 ./Nomad/monrepo/packages/multi-provider/tests/index.test.ts

353e4ad7d83f4a761fbab285ccc648020f8dd889bafff1c3d9e37e7d2dfbe930 ./Nomad/monrepo/packages/multi-provider/src/contracts.ts

5cdabb78281276666deb197ba2c892ad4bfd583594d831d375bd574240f1aa4b ./Nomad/monrepo/packages/multi-provider/src/domains.ts

04967c2ecc51f8911f6f46d16eff95361b45889b1499b657ee6d44df53724b10 ./Nomad/monrepo/packages/multi-provider/src/index.ts

8859d731870b0458a0bb80cb8497b0ecc4cba88023ad8e6bd38ff1df0ca7ee2b ./Nomad/monrepo/packages/multi-provider/src/provider.ts

b8452690bd44c5b217fa435bb93d101c211582b01300e6c64b4249d58c1e1a20 ./Nomad/monrepo/packages/multi-provider/src/utils.ts

cba229b2bc255137666c10cf76ce61af10d2cd5bf8480e9dbd2ae54ec1d9a067 ./Nomad/monrepo/packages/monitor/src/config.ts

f4190e3592a582e45a4f43b0646e2a1a73db8ab815d504240140117a9227c577 ./Nomad/monrepo/packages/monitor/src/gas.ts

6136be03dbae7aaec670bb9945f1e92fe23fa899a267e92d253aa7a8ef1d5f76 ./Nomad/monrepo/packages/monitor/src/googleSheets.ts

b707739d48342b764e748120a09feaf1110c59ae198f9b97666246c5e783b1e2 ./Nomad/monrepo/packages/monitor/src/metrics.ts

8f9c1418327c45d7fed3fcbe076200070be6ec54019b51c76b5cf56dabb5a1e0 ./Nomad/monrepo/packages/monitor/src/monitorSingle.ts

75aa8e6687b46b808a64c3241d9b9e084078167ac118cae5f611298dba94880e ./Nomad/monrepo/packages/monitor/src/print.ts

9ab796d8f6bf82a8894ebcb91a36d9f470f5d2cfd2dfc8865af6d1d7984c54e7 ./Nomad/monrepo/packages/monitor/src/registerContext.ts

817c104e4d29cf645280cb2983ee87b5d267036a4025f879ba3293250b3cf5ac ./Nomad/monrepo/packages/monitor/src/run.ts

11aa065a2ac8727f6059318d70d04f5e8eac4b4dfd5d41ff91552498bbc578e0 ./Nomad/monrepo/packages/monitor/src/setDetails.ts

30b26e8c452e4d02efad0256c8595350897a686a71861603705d8f547f4e66de ./Nomad/monrepo/packages/monitor/src/tokens.ts

0f32ca683e2f18120301d38569acc91ec37ff83eac23b746f8452392810b90d4 ./Nomad/monrepo/packages/monitor/src/trace.ts

31ca07e041fe5ef9bdef3118428508d3eddbb3398aefd84328d7b64642a7e73d ./Nomad/monrepo/packages/monitor/src/utils.ts

286a61390a5a804db89aa643d3cc8c7eb7665f4edb4ad8493318f1342360a374 ./Nomad/monrepo/packages/monitor/src/latencies/metrics.ts

45d0ee536f2f0ebcf03792afa543caa5fcd3fd3baaaef78a3bbb967692cd7349 ./Nomad/monrepo/packages/monitor/src/latencies/relayer/metrics.ts

de0d0d06bee65a7a4f8d3fc9a81cd2beb3dcd5c9c7485115f3955b7b18e7e406 ./Nomad/monrepo/packages/monitor/src/latencies/relayer/relayerMonitor.ts

0004a62efedbccd8bc831a8908537567b914d23434253ef547a1ace67aeedb57 ./Nomad/monrepo/packages/monitor/src/latencies/processor/metrics.ts

cfea02370f0fcdcbf3363da64b320032a4b30f2cf0d8b9f1c43b6f46854bb371 ./Nomad/monrepo/packages/monitor/src/latencies/processor/processorMonitor.ts

5fddd378b5b1c0e39dbf211555a10554cbd1e02cb1a4636d3f43826c07f1e822 ./Nomad/monrepo/packages/monitor/src/latencies/e2e/e2eMonitor.ts

ee46ee1437b8437c972fb2551209bfea9f1513528a478206e6b01862a94d601d ./Nomad/monrepo/packages/monitor/src/latencies/e2e/metrics.ts

5fa94435bbd2f7a37ae3f39e4c13ace752e6d1d25ff9cacf638aa47a9a9aaabe ./Nomad/monrepo/packages/monitor/src/bridgeHealth/healthMetrics.ts

84629728c9c4ed56161e76c203c0c77f3ee0357e5c4a4cfac8bfdb0050d4b94f ./Nomad/monrepo/packages/monitor/src/bridgeHealth/healthMonitor.ts

0a00a8da91dcd694e9f9f4d63b3327e75206441e581dbbfe96995b6699e08bbe ./Nomad/monrepo/packages/monitor/src/bridgeHealth/monitor.ts

58a12acc25827fb1b77e9717e4536c1554e4370a6ff7c81c83a751daad205516 ./Nomad/monrepo/packages/local-environment/tests/addChainCase.ts

3d305489b5c9d3b6eec418bd47c45b0ae6d85f5115a1219f138e7067bb5cdd6b ./Nomad/monrepo/packages/local-environment/tests/agentsDieOnImproperUpdate.ts

ed4a6ea2206e4db342cddd6ff2531225a2359cbf0159276b30fddf8fdf293a73 ./Nomad/monrepo/packages/local-environment/tests/common.ts

fee20b56627a4258e0c85c23f79282bda5c806925e6d8d019ef108a81bdd1cac ./Nomad/monrepo/packages/local-environment/tests/process.ts

9fa522ddb5a1596acf353b89007d199da93baac487a4a69e79e046c54acd0eea ./Nomad/monrepo/packages/local-environment/tests/sdkFailedHome.ts

e554b08520a21fb23c80c3de71f21e722a47b68515cc2de97b40dfb2759d2984 ./Nomad/monrepo/packages/local-environment/tests/sendTokensCase.ts

ceb917324b75a3512bd97c89c2ce12c3c437166caad6b063e54ad3755e6fdac0 ./Nomad/monrepo/packages/local-environment/tests/start.ts

7c29e2ade17c7d0c960cbde198a6e48add04315b2243c726cf1eed8ca34306df ./Nomad/monrepo/packages/local-environment/tests/startStopPersistanceCase.ts

ab9d6c95f45659011e35ba1812b6e9a3eb62856e55d6282a83e5f179c2b055d3 ./Nomad/monrepo/packages/local-environment/tests/stop.ts

0d9add37c151aec58fa74cc16ec68236355ba35119c3311aa906152f2a94eb3b ./Nomad/monrepo/packages/local-environment/tests/watcherDoubleUpdate.ts

6bf0829e5403caca767fc30758b018056938ef638bf9f1c7f890dc0f54813490 ./Nomad/monrepo/packages/local-environment/tests/watcherImproperUpdate.ts

f1287caa4231823c32507842ba1040ad849c1eda8fdb2ebe3b4896b759463e62 ./Nomad/monrepo/packages/local-environment/tests/utils/token/deployERC20.ts

eb0167b1c14cef3031e76e798268da52fd19d43c30331f502f95bc5d5ad252f3 ./Nomad/monrepo/packages/local-environment/tests/utils/token/contract/Context.sol

60c3bb830c82a94e21fc9e4ece1400282ba46d1244afd251404735cbb6e72825 ./Nomad/monrepo/packages/local-environment/tests/utils/token/contract/ERC20.sol

6a7b7a3f5b0f0011b42bb3ae925ada6b06a24633c63b38d8bce6234a7313f034 ./Nomad/monrepo/packages/local-environment/tests/utils/token/contract/IERC20.sol

8e51e13ae9e0df1dd2336d8f7a20620bc3551e1f5719af4eabe48938684c127a ./Nomad/monrepo/packages/local-environment/tests/utils/token/contract/Token.sol

f7e87d55d7c593a9837685496939d6690e228f54ba79158d5f2024e46a0545f3 ./Nomad/monrepo/packages/local-environment/src/actors.ts

f8bddd5fe59f0a2d7cdc355bdd8cee7e906d2a7224ddfde5f34e43fd8a5efac6 ./Nomad/monrepo/packages/local-environment/src/agent.ts

9ae920edaab621727e0121453095dc15e4850da5b0a72df82fce8f30afe91832 ./Nomad/monrepo/packages/local-environment/src/index.ts

dc088a194fc09bf0b4105268fbc18c6dfcc0e0e6342e130baa739af57ed3892a ./Nomad/monrepo/packages/local-environment/src/key.ts

7f22f816304033da3ceb8d7a47ab4743183458fc2990a985aafc91dfe6923bcc ./Nomad/monrepo/packages/local-environment/src/logger.ts

f60bae1ce22b505e61a105c03c6ac776c9387046bd00435c529a035f7806eaf6 ./Nomad/monrepo/packages/local-environment/src/network.ts

6b729140745855333ab7ce3972144f7d3bf3dc8e7363c048cc55a20d9dd7ea1b ./Nomad/monrepo/packages/local-environment/src/nomad.ts

dd460b0fcbbfe1be6d9592f174dcb58e8057a032a00aad2c099622fb3f3b6a6e ./Nomad/monrepo/packages/local-environment/src/types.ts

125929a21dc455e6a2da9fff8c4c13ceeca3862afd4e52130ed9130521389a2a ./Nomad/monrepo/packages/local-environment/src/updater.ts

f73bce57d59f70021e430352fd6bd037673db0786b45c21a0a2eaaee4eb0871c ./Nomad/monrepo/packages/local-environment/src/utils.ts

41a82e8c191977647a168099de1248929d8e99e3f766594bbf30ef9bc941444b ./Nomad/monrepo/packages/local-environment/hardhat/hardhat.config.js

e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 ./Nomad/monrepo/packages/keymaster/__init__.py

d0b215b38548c6fe92fda8dccfdd91c4dc72fb6b8c6f6103a1f1f4d276f3045b ./Nomad/monrepo/packages/keymaster/src/config.py

e4b1fb57cd2a115f52e57952c596580f23a89da58537c233bacfb5901f7e9098 ./Nomad/monrepo/packages/keymaster/src/keymaster.py

6359cedcb1383ac2c9df0374c5b76713ccf01fcd450dec9caf6ff74dda9a14ae ./Nomad/monrepo/packages/keymaster/src/utils.py

e3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855 ./Nomad/monrepo/packages/keymaster/src/__init__.py

21e097e8c2837b5653bf81f5bf42c6d284e17c5b8685604df1a41c574fc2bc12 ./Nomad/monrepo/packages/indexer/main.ts

5e4aee63cff36bc604430a50fd9089b4bae5f7ae20f3be0640621fdf99ce560a ./Nomad/monrepo/packages/indexer/tools/kek.ts

91f02471a521428f0f5e7088c4f71ca0755a4e32340372f8f9e238abb5737400 ./Nomad/monrepo/packages/indexer/tools/lol_logs.ts

fc6aed73bd80291c56213adcc03676235fcab120a2d6f9222ae5fbacc74315ab ./Nomad/monrepo/packages/indexer/tools/split_logs.ts

85ccb6f1c28519834924b4b9d1ca48e1a0d366ab2ce7417dd52112eac043524f ./Nomad/monrepo/packages/indexer/core/api.ts

efec906537730ef987bce9c28f27e47da7c757d3daa1ac0221acb233117d1ec9 ./Nomad/monrepo/packages/indexer/core/consumer.ts

3f8bcb8f29a18b9cc0647cad2c96f090db353d2f76c2f512a856d52d121a05fe ./Nomad/monrepo/packages/indexer/core/db.ts

9af0b71e628c58ca8585d07294703c89b0a27faef301c102d18fcbc5baf95dd2 ./Nomad/monrepo/packages/indexer/core/event.ts

bda0114a991d7cd9262cfd2d03da1562997f37791ea6498d205840d36348f0db ./Nomad/monrepo/packages/indexer/core/index.ts

0257db4737e5209bf95c7f1fd708c0eb58d17f4f1003f6df1f8526532fe2d702 ./Nomad/monrepo/packages/indexer/core/indexer.ts

c6288a09095ced16803a00aed43dded4c512fd08ef41e62fab6994e6d76f83ee ./Nomad/monrepo/packages/indexer/core/metrics.ts

468fee0e637bb251e1f9bd6c4890cdc5becaf47f641d244a46ae8bf1e7344afd ./Nomad/monrepo/packages/indexer/core/orchestrator.ts

abbfb12547c2082c2c15ede5f7f1b72ffb97f92abecaf15341dc1d038b342435 ./Nomad/monrepo/packages/indexer/core/types.ts

ad77b85d8ebc18570582fa4926ccad27055445f76bc4c69aaa3605a3f56f291c ./Nomad/monrepo/packages/indexer/core/utils.ts

956eca308a3d91cb5ef77ad0fc968b92aaf73934be50f671bb79f307dc434229 ./Nomad/monrepo/packages/indexer/api/index.ts

de2c17b01608a759cb09474dec14727b81a29e6489749d95523202f09fef8db0 ./Nomad/monrepo/packages/deploy/src/chain.ts

51791620a3884698c0d93d841a33d452ce7ff5dcdf9039b106daba57cfb530c3 ./Nomad/monrepo/packages/deploy/src/contracts.ts

76c08759f1c68a3604ad96de34db81753805e87f5ce12e0ee565331feec7410c ./Nomad/monrepo/packages/deploy/src/deploy.ts

9f15c19fb2931f160d5b000c11939f4422bb4d2a95bbe2f02c09397b15bad731 ./Nomad/monrepo/packages/deploy/src/index.ts

f3c151e8de531fa353c3e86dbbb71f1338159ea9fa77fd7ac76cfd1241d12d8b ./Nomad/monrepo/packages/deploy/src/proxyUtils.ts

87c685ccb18a3f897d7e4b2f22a8faefb94de7d4b9e420f07eb1ecb403133d98 ./Nomad/monrepo/packages/deploy/src/utils.ts

55eb4222d357afd770942f850ad36193be915d079e982947d104fc2686359f32 ./Nomad/monrepo/packages/deploy/src/verification/readDeployOutput.ts

9b4a44d88f5b02bfeced80eebac6f9b5e915173de1e6bd229b1c562709704640 ./Nomad/monrepo/packages/deploy/src/verification/verifyDeploy.ts

e31c9e744359af969f734cfbaae8329ea994e3fcb7981a4c9a7ed2baf23206aa ./Nomad/monrepo/packages/deploy/src/verification/verifyProxy.ts

3ba5f5975844e276fa47ab97fef973ec24bd1fbf095d4972d56c278232351b2e ./Nomad/monrepo/packages/deploy/src/incremental/checks.ts

017db3ac596d480ecdc229baecb13b21bd2763bc3f7e04a3226bd4d76295c39d ./Nomad/monrepo/packages/deploy/src/incremental/index.ts

e0e0ee512dec3b533c9f20a6c322586288e28380625e78b73fcdf9be626ab2f1 ./Nomad/monrepo/packages/deploy/src/incremental/utils.ts

83cc299369f2146a67a55bb0705dfc77f76ad97dd3bd9bbaf6cee93240526b74 ./Nomad/monrepo/packages/deploy/src/core/checks.ts

9e97868cdd02b1b8f5c2487bc9ffed00699a3777adf43a97942995ee4b279234 ./Nomad/monrepo/packages/deploy/src/core/CoreContracts.ts

1c80d2f5fce45dca3bb3e6436f62191c6ea7850160366978c773785bc3881cb5 ./Nomad/monrepo/packages/deploy/src/core/CoreDeploy.ts

eb3d840e88b3e621fdc7bc717c112ae80e368e7558da6a1df94b64ad28eed740 ./Nomad/monrepo/packages/deploy/src/core/index.ts

7ae32ae48b0ccda2e146a6eebd5e613419a0dafc52db021114339c32c44725d2 ./Nomad/monrepo/packages/deploy/src/bridge/BridgeContracts.ts

a0318b34b2cee9c920317197bf064ecb09c40e144ecea4408edc6ec581e4945c ./Nomad/monrepo/packages/deploy/src/bridge/BridgeDeploy.ts

588ed3a6cddf7dcf0125a68d0f97dce9be07d843d4d4ffc0337677b16758b38f ./Nomad/monrepo/packages/deploy/src/bridge/checks.ts

290f444ec84324f8de7e89ca6ed197f9cf1035dd77b63f0188f0a0cdcc4ed7e5 ./Nomad/monrepo/packages/deploy/src/bridge/index.ts

8fcd05796972a6ff8b2bdacf1ed35d0df9ab4fb93a1f8c9df4df64f2f3842b98 ./Nomad/monrepo/packages/deploy/scripts/calculateDomain.ts

ea76c39fae46bdd5283209e4f8715e3a53e5b36639d097fc1253ccd605d4c3af ./Nomad/monrepo/packages/deploy/scripts/staging/bridge.ts

18757af12315b2a425cd8885f4e2016d0484f0e6c8f81e40237a00d83ab6f208 ./Nomad/monrepo/packages/deploy/scripts/staging/core.ts

cdf634eb7d270f89e56c14ca867453089da3669c208ffaddf3d052493830fa3a ./Nomad/monrepo/packages/deploy/scripts/staging/governance/checkNewChain.ts

dd41c44e4f9c48f7836e17b08a00fe0aaa5454270dd45bc94fa8745818c9a7f4 ./Nomad/monrepo/packages/deploy/scripts/staging/governance/deployNewBridge.ts

91ef60674514ea792b196d20c642c62ea32bda3b037fce6d8721e0bd89d250fb ./Nomad/monrepo/packages/deploy/scripts/staging/governance/deployNewCore.ts

7dfefeb82fc76ef37c971915155d2db076252f223490967b75c48e08b37eb3a7 ./Nomad/monrepo/packages/deploy/scripts/staging/governance/enrollNewChain.ts

a0d4bbb0e2ad5aa419a048d3ffdc5d08b5b1de1f1bc4c066d50a69929ed698d7 ./Nomad/monrepo/packages/deploy/scripts/mainnet/bridge.ts

7776da155405407bbd6029da329df1a5ef37d2e051bb1f2e68e610ab29b9511a ./Nomad/monrepo/packages/deploy/scripts/mainnet/core.ts

421b7e80e39b67dc2f15bb281384ad90280b398e095fa4921cf4cf799a9b6b65 ./Nomad/monrepo/packages/deploy/scripts/local/addBridge.ts

c06d307db38682a898480a70407b88b97a01638b56c80155e599c9a43ce37872 ./Nomad/monrepo/packages/deploy/scripts/local/addCore.ts

71a0691f6427f4f604301910af5516d3583c7c3b3e82d994e807a16ff2cefcbf ./Nomad/monrepo/packages/deploy/scripts/local/bridge.ts

3b5d1cf8a175ce41aed5f8b89e56e7e6d6121907038d78d5549733bb8789929e ./Nomad/monrepo/packages/deploy/scripts/local/connect.ts

26d08cf9a93a6bee659ab2db7500ea438ad4c4e521d0371bbc9931318ea446a6 ./Nomad/monrepo/packages/deploy/scripts/local/core.ts

1db51f5e76654c79b48fb8a9e2ed069b0cfea0b138c8e783e66fd47899a33b71 ./Nomad/monrepo/packages/deploy/scripts/development/bridge.ts

91bb0bfd176e1a72f31220d9ecf0bc1a7d8b01d0a7ee1933ef3ffdaa2a558601 ./Nomad/monrepo/packages/deploy/scripts/development/core.ts

07bf32de7964b398a1b72445fe7d46be18d1076e9eb975ecc5e4e6f67a842d51
./Nomad/monrepo/packages/deploy/scripts/development/governance/checkNewChain.ts

b6e9c2ec8a1964928b2c11b5274803ac9b0b3d07a6fe1d6961e83c54176c5ee0
./Nomad/monrepo/packages/deploy/scripts/development/governance/deployNewBridge.ts

5465d6b32c1f6ed0f544807a1241dde05b62ce92c57fbd827b8a8e5e3d10a11d
./Nomad/monrepo/packages/deploy/scripts/development/governance/deployNewCore.ts

2ad1ce919c7f984924612f962266eb751d49169e28f555d38b17efdf035c6cf2
./Nomad/monrepo/packages/deploy/scripts/development/governance/enrollNewChain.ts

2e6a2a208c2cc279e70bb2b283f7c930dca23dd0e75de87b2e645c8909e8f456 ./Nomad/monrepo/packages/deploy/config/testnets/kovan.ts

2591abfd8c453663952a41740c7d01f27d5b3e8919c97f2104c2d716047fc83a ./Nomad/monrepo/packages/deploy/config/testnets/milkomedaTestnet.ts

8995fc4a24ac095b39071637054c6b39d9dfd1d9c8699ca30fef8c2c398eabcb ./Nomad/monrepo/packages/deploy/config/testnets/moonbasealpha.ts

7485e8cb0daf2b285351bd08137bb590da5c5deaaf1479c0ac6486bb1c4cd125 ./Nomad/monrepo/packages/deploy/config/testnets/rinkeby.ts

4470c89496416da5cced13878cf62709ca195ebc0592c34e993568d679c7ae00 ./Nomad/monrepo/packages/deploy/config/mainnets/astar.ts

03bf5f52df90584decc9a0d1237c148ad1ad9d17fc35c8ced7dd6279ea7d755e ./Nomad/monrepo/packages/deploy/config/mainnets/ethereum.ts

48104c0dad6191c2aab62dfb10e9ada10e5e8259eeb4d375b983a2d94b575e65 ./Nomad/monrepo/packages/deploy/config/mainnets/moonbeam.ts

9260d314dd584c910bfc480dad5e4e4e05f1dd1f831c7999f55eda3a81c90e70 ./Nomad/monrepo/packages/deploy/config/local/daffy.ts

5ac11dc1333273724df61cdc5d836be260f0bbc5e3971038830fca910981f434 ./Nomad/monrepo/packages/deploy/config/local/jerry.ts

f69baf8aa2147dd8271d29b776b76158da9ba3cc41e10ae933fd383474e4c245 ./Nomad/monrepo/packages/deploy/config/local/tom.ts

7e67c8663d9d8ab563181f3a93b2e9f7f6e04d5fbb4a0c91f46e3131b2114832 ./Nomad/monrepo/packages/contracts-router/.solcover.js

ba60b1ae55c84d0acef6b5e99330f0c8630acca4546ad93bb12c3f4fe2c6e3ae ./Nomad/monrepo/packages/contracts-router/hardhat.config.ts

652b3852ad8e63efd4f91567fd4a004a4e939766e0ecd5bc3951fd58aa3442de ./Nomad/monrepo/packages/contracts-router/index.ts

dcbde8a7a9fe5b98888fee130a7f3d7e163b92ad0928f10580aef6b147b01954 ./Nomad/monrepo/packages/contracts-router/dist/index.d.ts

e49c87326bc40b0b1c0df9e72a7edf7c42b23c1b133218a926c82ddd03904368 ./Nomad/monrepo/packages/contracts-router/dist/index.js

b6b3e1a64bb480a8315a55946678c8e290412f5e3ed817f8bae9e819d86f8960 ./Nomad/monrepo/packages/contracts-router/contracts/Router.sol

018f5299b11762b2d2bdb3274fe11f6ec9dbfa8295f65fce5e7d97b9350f8f18 ./Nomad/monrepo/packages/contracts-router/contracts/XAppConnectionClient.sol

7e67c8663d9d8ab563181f3a93b2e9f7f6e04d5fbb4a0c91f46e3131b2114832 ./Nomad/monrepo/packages/contracts-core/.solcover.js

ce15333780839d48856bde1178279418164502b8186c8f280827c458bf71cf3a ./Nomad/monrepo/packages/contracts-core/hardhat.config.ts

f9bf3bb9e8b2160082395589a791c990e1bd8e3e49d3aa416a4886c0960394fc ./Nomad/monrepo/packages/contracts-core/index.ts

65b89b66f7bca650351992b273ad5c23f23d075f45907f53943461f2be68f6a9 ./Nomad/monrepo/packages/contracts-core/dist/index.d.ts

6c49c717cb1cb02315a51a4f2a190f6d366c8c78f956b02cc733c488e352190a ./Nomad/monrepo/packages/contracts-core/dist/index.js

d00a117e25bc8ba1071845d331028a3c58cb5dd5d1e42a2f8571c425aef2196d ./Nomad/monrepo/packages/contracts-core/contracts/Home.sol

d0f40e08a0ed29e6fb3e208a412a17f2c07db17a558dac7091c61686088ed5c3 ./Nomad/monrepo/packages/contracts-core/contracts/Merkle.sol

de15ff842258e7e22af1bfd2183a86f3d183bd6f416040c290e7c2f4966718a9 ./Nomad/monrepo/packages/contracts-core/contracts/NomadBase.sol

```
7fb0fd531a9f2283ef1cbd3c24e2d0fa3feb8b1ddc928e19f53e4544fe031a41 ./Nomad/monrepo/packages/contracts-core/contracts/Queue.sol

4b9de559b63b90a12907be0931e8769ee406642312574e52d3349a71355e1831 ./Nomad/monrepo/packages/contracts-core/contracts/Replica.sol

d1ad5ebafc193db8f0723b9e38b0c9c35a89c3739cab44c6fa88fd2247b0477f ./Nomad/monrepo/packages/contracts-core/contracts/UpdaterManager.sol

e2b284e9446b30386024b9aa0e1500488a3fc7d0e8aa747a45aa4fe6590263a3 ./Nomad/monrepo/packages/contracts-core/contracts/Version0.sol

f6ef802dd25c3abed7fe87c00c7692e286cba4b608ad2eadb4dd65733ee81bb1 ./Nomad/monrepo/packages/contracts-core/contracts/XAppConnectionManager.sol

1c0e101260c2549940880292443e755989cf8c61b871bc66b6626b247503965d ./Nomad/monrepo/packages/contracts-core/contracts/upgrade/UpgradeBeacon.sol

832ffa316e24d689c19d7023748a267712d84d3cf6a7e3f90289be808cebad15 ./Nomad/monrepo/packages/contracts-
core/contracts/upgrade/UpgradeBeaconController.sol

6bf68a5b556dac76487fb922a5c5ba1a373e67f57368dce683e8ce07619d79d8 ./Nomad/monrepo/packages/contracts-
core/contracts/upgrade/UpgradeBeaconProxy.sol

1d8f79d86f912ed20c44266a7296efb01ebe466af521d2bbd4b3beb990567dd3 ./Nomad/monrepo/packages/contracts-core/contracts/test/MysteryMath.sol

4ea99fa2bf1e686dd296d71e5c358c7b306ec926b9fcfcf2b50214058a242adf ./Nomad/monrepo/packages/contracts-core/contracts/test/MysteryMathV1.sol

47adc30c71b54462030de2e089c102ac27fb0de682f4921bd8f6559499abff38 ./Nomad/monrepo/packages/contracts-core/contracts/test/MysteryMathV2.sol

7f6beb01d945d5d32901f21ad8f52ba8dbe932e10d68e6580a2e3e7eac8e4503 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestCommon.sol

f30ca27e325e08cfec07a382dc2cb41512c4d1832906c4d390ae59e5911e969f ./Nomad/monrepo/packages/contracts-
core/contracts/test/TestGovernanceMessage.sol

2a93462dde81dc77165785e9c7a780e14ba44d93ebbcb0ae9ec545eb507ddb6e ./Nomad/monrepo/packages/contracts-
core/contracts/test/TestGovernanceRouter.sol

f2ee9ecb9a7717f6c59283b076fe779450092118015915e4edc6e7935b7095f3 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestHome.sol

668a6c81374b08925970d8ed2c09b15f044c56c51b45e33a7a2d98f5c93605f0 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestMerkle.sol

0cea4931ae64899e37f4282343555240ebd51a77c24dd6942903d75413219276 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestMessage.sol

8192727c23fb18126efbe62c3901164ac468ff8d98b3cabdfedb74ec7345540e ./Nomad/monrepo/packages/contracts-core/contracts/test/TestQueue.sol

63f5b9ef69d92b9b8fd50d08040ea95cc6fcaee3dc1f21a38b4fdfbc80404b33 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestRecipient.sol

87c75a1dfd272782de15bd380bc650c6fc05aef8f5621f770eee2ede492a7bb0 ./Nomad/monrepo/packages/contracts-core/contracts/test/TestReplica.sol

7fe0ca4b7ac0666f1625ef65159a338f2943cabe7b6668e7f4844d30081eee04 ./Nomad/monrepo/packages/contracts-
core/contracts/test/TestXAppConnectionManager.sol

32dcd82fa0e32e3745eca4f9e39458eee1fb0683d22d3dc6b6b99638a3aabcb2 ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient1.sol

df57b1d5fc8f518c0d13ad72894445b8bf4efc45c13c540289aed9055d911a99 ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient2.sol

a252945415af04654838bdc3959c979219965eb31ab88016cf7c624a5e4b0e4c ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient3.sol

3ba8e0c4f42fa62321b94b58756488851b5c365ed4722ca306d4bc265b1385c4 ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient4.sol

e9d43d25bc487b1b8a5335adeec84197bcf16d0910be6f30740333bd258a98f4 ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient5.sol

2ebbec7b7fc89f2d49dc473cbedf6a7f1f54abd359fc3f78317062497b17e636 ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipient6.sol

63b9010a994b096ea2d787b38921b0f995543b15f19f07a6c33e5c1a90eafeaf ./Nomad/monrepo/packages/contracts-core/contracts/test/bad-
recipient/BadRecipientHandle.sol

ba936cbe96f54b719623b6c95a2f069356fe14e8061196d9f02884ddf1deffc8 ./Nomad/monrepo/packages/contracts-core/contracts/libs/Merkle.sol

e2262f0517d1d69c5d200b3ac6998134ca24ff4737b89e0514995545 9f650c67 ./Nomad/monrepo/packages/contracts-core/contracts/libs/Message.sol

25bf0b13e085ab095235ff25bf842b5b2b03d0992afc4aec2cec8d3ae2403b60 ./Nomad/monrepo/packages/contracts-core/contracts/libs/Queue.sol

b82274ebdf155c9cd1750a64b864a54c17c79cf59188b7c58e33f3fe2c09e21c ./Nomad/monrepo/packages/contracts-core/contracts/libs/TypeCasts.sol

40479a0d3596ddcd00fa2ff6e24cee120617e784bca4ec4413e461e383ae89e4 ./Nomad/monrepo/packages/contracts-
core/contracts/interfaces/IMessageRecipient.sol

7958f62ba4adb2d6367d94a8736585189e7b70b7ff5794fc22770a267c35b9a1 ./Nomad/monrepo/packages/contracts-
core/contracts/interfaces/IUpdaterManager.sol

a165f6a668b15714bccb4b495372ddf87d68cc5d21f8481cb27475cfddca4620 ./Nomad/monrepo/packages/contracts-
core/contracts/governance/GovernanceMessage.sol

7d80a2fd57a7dd4bf28b858b8552c355a32629dcb8a156ddcee8b7f2b4bec2c8 ./Nomad/monrepo/packages/contracts-
core/contracts/governance/GovernanceRouter.sol

f9567ce10c06ba7449f98e65c78f0a6f9ae72ee993c88a1875fa6a92a7ceeaa4 ./Nomad/monrepo/packages/contracts-bridge/hardhat.config.ts

505a1adc05c826c60f424bcf08690f87ce4c39472e31fe2c98a0425b878aec77 ./Nomad/monrepo/packages/contracts-bridge/index.ts

de77141ff9209d1689da46043d920e9a3ad008b5a3924355244aaff814e2dabd ./Nomad/monrepo/packages/contracts-bridge/dist/index.d.ts

78fc038def97c4ca7114ce97cd4608263c0e293c8029fe93f2fefb8bb22175d9 ./Nomad/monrepo/packages/contracts-bridge/dist/index.js

a06c0a999cf1d52ef4afe30a315241c33c3d711c164897e1c94fd78e86d7502c ./Nomad/monrepo/packages/contracts-bridge/contracts/BridgeMessage.sol

70bfd5e9872f2d5c69e97d9bc00c487460a1643000888aeef8ae5ecd5a49ee59 ./Nomad/monrepo/packages/contracts-bridge/contracts/BridgeRouter.sol

471e85f1fb02b5fb766bee82ff222cffd6d1c520e4bf477b07a653633a43122a ./Nomad/monrepo/packages/contracts-bridge/contracts/BridgeToken.sol

8e5faf5a8fb075aec05853c8c40e1cc258de4adeb4b85f9edd646a6bed1c5c45 ./Nomad/monrepo/packages/contracts-bridge/contracts/Encoding.sol

fd1783d494ee70cff5b0015e6b743792b35d92d4516e23199a07f4a7545a3cbd ./Nomad/monrepo/packages/contracts-bridge/contracts/ETHHelper.sol

a730af6c84a6774f449fd2516b0be0c40cb5efd43f33d803c8c5530bc54d82a1 ./Nomad/monrepo/packages/contracts-bridge/contracts/TokenRegistry.sol

c9f0d8defff61d9f6268b0d198f9aa64c687d1659dc38b134c46c9ee5e43a93 ./Nomad/monrepo/packages/contracts-bridge/contracts/vendored/OZERC20.sol

cba19a6b53ed08db707391b2998e8cdcb7141357738d37a677e51bf07e9eba4d ./Nomad/monrepo/packages/contracts-bridge/contracts/test/MockCore.sol

eae58a9770879c46b0fac491d837e2def75f6646faa98e93639b2d7a9ab65dec ./Nomad/monrepo/packages/contracts-bridge/contracts/test/MockWeth.sol

dcf6ab9f802fa128422be1a18798478ae81f2f552a17888d49a596f2a3aeb1e9 ./Nomad/monrepo/packages/contracts-
bridge/contracts/test/TestBridgeMessage.sol

abb23624cc9441d042cd697d263a1cd96ef5109a9ea05deb2e8366c5bb8448e7 ./Nomad/monrepo/packages/contracts-bridge/contracts/test/TestBridgeRouter.sol

ea8d6853a57b3b86065bdf7a59dd271922d9c25e296c5cac00e7b75ef0414e0b ./Nomad/monrepo/packages/contracts-bridge/contracts/test/TestEncoding.sol

e70586fb593fd2da4827c1e4d0171c213d83af2f87761fdf53dbc51333860e80 ./Nomad/monrepo/packages/contracts-
```

```
bridge/contracts/interfaces/IBridgeToken.sol

28ff236d43f9be5f244554759ce1f710a6ff8c69916b74d20f400e5816a3fbb6 ./Nomad/monrepo/packages/contracts-
bridge/contracts/interfaces/ITokenRegistry.sol

6601834c30605d07982020a1f0788824500f55278c528ead3721f97a429dffd5 ./Nomad/monrepo/packages/contracts-bridge/contracts/interfaces/IWeth.sol
```

Changelog

- 2022-05-11 - Initial report
- 2022-06-06 - Re-Audit report

About Quantstamp

Quantstamp is a Y Combinator-backed company that helps to secure blockchain platforms at scale using computer-aided reasoning tools, with a mission to help boost the adoption of this exponentially growing technology.

With over 1000 Google scholar citations and numerous published papers, Quantstamp's team has decades of combined experience in formal verification, static analysis, and software verification. Quantstamp has also developed a protocol to help smart contract developers and projects worldwide to perform cost-effective smart contract security scans.

To date, Quantstamp has protected $5B in digital asset risk from hackers and assisted dozens of blockchain projects globally through its white glove security assessment services. As an evangelist of the blockchain ecosystem, Quantstamp assists core infrastructure projects and leading community initiatives such as the Ethereum Community Fund to expedite the adoption of blockchain technology.

Quantstamp's collaborations with leading academic institutions such as the National University of Singapore and MIT (Massachusetts Institute of Technology) reflect our commitment to research, development, and enabling world-class blockchain security.