Report

v. 3.0

Customer

Blockswap

Smart Contract Audit

Modular Gateway

12th June 2023

Report prepared by

ABDK Consulting

Contents

1 Changelog

#	Date	Author	Description
0.1	24.05.23	A. Zveryanskaya	Initial Draft
0.2	24.05.23	A. Zveryanskaya	Minor revision
1.0	25.05.23	A. Zveryanskaya	Release
1.1	02.06.23	A. Zveryanskaya	Scope update
2.0	02.06.23	A. Zveryanskaya	Release
2.1	02.06.23	A. Zveryanskaya	Introduction update
3.0	12.06.23	A. Zveryanskaya	Release

2 Introduction

The following document provides the result of the audit performed by ABDK Consulting (Mikhail Vladimirov and Dmitry Khovratovich) at the customer request. The audit goal is a general review of the smart contracts structure, critical/major bugs detection and issuing the general recommendations.

Blockswap Labs is a research and development firm making blockchain technology accessible to mainstream users. As core contributors to Blockswap Network and Proof of Neutrality Network, Blockswap Labs are building a permissionless middle layer and catalyzing web3 development through credibly neutral public benefit infrastructure solutions.

3 Project scope

We were asked to review several repositories:

- Original Code
- Code with Fixes

Files:

m-g/		
	Gateway.sol	GatewayToken.sol
/m-g/consent/		
	ConsentVerification.sol	
m-g/errors/		
	CommonErrors.sol	
m-g/t-m/dispenser/		
	Recovery.sol	
m-g/t-m/dispenser/dETH/		
	dETHDestination Dispenser.sol	dETHOrigin Dispenser.sol
m-g/t-m/dispenser/ERC20/		
	ERC20Destination Dispenser.sol	ERC20Origin Dispenser.sol
m-g/t-m/ingestor/dETH/		
	dETHDestination Ingestor.sol	dETHOriginIngestor.sol
m-g/t-m/ingestor/ERC20/		
	ERC20Destination Ingestor.sol	ERC20OriginIngestor.sol

m-g/accumulator/		
Accumulator.sol	AccumulatorFactory.sol	
m-g/g-i/dETH/		
savETHDestination Gateway.sol	savETHDestination Reporter.sol	savETHGateway.sol
savETHOrigin Gateway.sol	savETHRegistry DestinationGateway.sol	
m-g/g-i/dETH/		
StakeHouse UniverseAPI.sol	StakeHouse UniverseDestination Gateway.sol	
m-g/g-i/ERC20/		
ERC20Gateway.sol		
m-g/interfaces/		
IAccountManager.sol	IAccumulator.sol	IAccumulator Metadata.sol
IEIP721Signature.sol	IGateway.sol	IGatewayDispenser.sol
IGatewayIngestor.sol	IGatewayOperator ControlCentre.sol	IGatewayStructs.sol
IRBSEndorser Signature.sol	IsavETHDispenser.sol	ISavETHManager.sol
ISavETHRegistry.sol		
m-g/RPBS/		
EndorserRegistry.sol	RPBSVerification Library.sol	

4 Methodology

The methodology is not a strict formal procedure, but rather a selection of methods and tactics combined differently and tuned for each particular project, depending on the project structure and technologies used, as well as on client expectations from the audit.

- **General Code Assessment**. The code is reviewed for clarity, consistency, style, and for whether it follows best code practices applicable to the particular programming language used. We check indentation, naming convention, commented code blocks, code duplication, confusing names, confusing, irrelevant, or missing comments etc. At this phase we also understand overall code structure.
- **Entity Usage Analysis**. Usages of various entities defined in the code are analysed. This includes both: internal usages from other parts of the code as well as potential external usages. We check that entities are defined in proper places as well as their visibility scopes and access levels are relevant. At this phase, we understand overall system architecture and how different parts of the code are related to each other.
- **Access Control Analysis**. For those entities, that could be accessed externally, access control measures are analysed. We check that access control is relevant and done properly. At this phase, we understand user roles and permissions, as well as what assets the system ought to protect.
- **Code Logic Analysis**. The code logic of particular functions is analysed for correctness and efficiency. We check if code actually does what it is supposed to do, if that algorithms are optimal and correct, and if proper data types are used. We also make sure that external libraries used in the code are up to date and relevant to the tasks they solve in the code. At this phase we also understand data structures used and the purposes they are used for.

We classify issues by the following severity levels:

- **Critical issue** directly affects the smart contract functionality and may cause a significant loss.
- **Major issue** is either a solid performance problem or a sign of misuse: a slight code modification or environment change may lead to loss of funds or data. Sometimes it is an abuse of unclear code behaviour which should be double checked.
- **Moderate issue** is not an immediate problem, but rather suboptimal performance in edge cases, an obviously bad code practice, or a situation where the code is correct only in certain business flows.
- **Minor issues** contain code style, best practices and other recommendations.

5 Our findings

We found 1 critical, 19 major, and a few less important issues. All identified Critical issues have been fixed.

Issues	Active	Fixed
Severity: Critical	0	1
Major	16	3
Moderate	6	1
Minor (no security issues)	209	0

Fixed 5 out of 236 issues

6 Critical Issues

CVF-1. FIXED

- **Category** Flaw
- **Source** Gateway.sol

Recommendation Should be ”_isEnabled” instead of ”true”.

```
702 isGatewayAdmin[_gatewayAdmin] = true;
```

7 Major Issues

CVF-2. FIXED

- **Category** Unclear behavior
- **Source** Gateway.sol

Description The ERC-165 standard suggest to first ensure that a contract does support ERC-165 API, before using it.

Recommendation Consider checking ERC-165 support as described in the standard: https://eips.ethereum.org/EIPS/eip-165#how-to-detect-if-a-contract-implements-erc-165

```
219 if (
220     !IERC165(address(_domainMetadata.ingestionModule)).
          ↪ supportsInterface(
            type(IGatewayIngestor).interfaceId
        )
    ) revert InvalidIngestorInterface();
```

```
225 if (
        !IERC165(_domainMetadata.dispenserModule).supportsInterface(
            type(IGatewayDispenser).interfaceId
        )
    ) revert InvalidDispenserInterface();
```

CVF-3. INFO

- **Category** Unclear behavior
- **Source** Gateway.sol

Description Zero environment ID is implicitly forbidden, as it would cause underflow.

Recommendation Consider either handling zero environment properly, or explicitly forbidding it.

Client Comment *Disagree with this. Chain zero is not a possibility. Unless there is an EVM chain with ID zero, we cannot acknowledge this.*

```
482 sha256(abi.encode(_environmentId - 1))
```

CVF-4. INFO

- **Category** Suboptimal
- **Source** EndorserRegistry.sol

Description The expression ”lifecycleStatusOfEndorserByPublicKey[pubKey]” is calculated several times.

Recommendation Consider calculating once and reusing.

Client Comment *As this does not break anything, we see it as a future optimisation.*

```
65 lifecycleStatusOfEndorserByPublicKey[pubKey] ==
      ↪ EndorserLifecycleStatus.ACTIVE ||
   lifecycleStatusOfEndorserByPublicKey[pubKey] ==
      ↪ EndorserLifecycleStatus.KICKED,
```

```
72 uint256(_status) - uint256(lifecycleStatusOfEndorserByPublicKey[
      ↪ pubKey]) >= 1,
```

CVF-5. INFO

- **Category** Suboptimal
- **Source** Accumulator.sol

Description Zero hashes are effectively constants. Keeping them in the storage is expensive.

Recommendation Consider just using zeros instead of zero hashes.

Client Comment *We want to avoid modifying code from original Ethereum deposit contract so don’t wish to change this.*

```
32 bytes32[TREE_DEPTH] public zero_hashes;
```

CVF-6. INFO

- **Category** Suboptimal
- **Source** dETHOriginIngestor.sol

Description The expression ”universe.saveETHRegistry()” is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *As this does not break anything, we see it as a future optimisation.*

```
68 uint256 associatedIndexId = universe.saveETHRegistry().
      ↪ associatedIndexIdForKnot(_blsPublicKey);
   uint256 knotDETHBalanceInIndex = universe.saveETHRegistry().
      ↪ knotDETHBalanceInIndex(associatedIndexId, _blsPublicKey);
```

CVF-7. INFO

- **Category** Bad naming
- **Source** Recovery.sol

Description Semantics of the arguments is unclear.

Recommendation Consider giving descriptive names to the arguments and/or describing in a documentation comment.

Client Comment *We will aim to add documentation. These params are generic to support multiple asset types in the future.*

```
159 function _performDispense(uint256, address, uint256, bytes calldata)
       ↪  internal virtual returns (uint256);
```

CVF-8. INFO

- **Category** Readability
- **Source** dETHDestinationDispenser.sol

Description Using business-level properties to check technical-level constraints, such as whether a contract was initialized, is a bad practice.

Recommendation Consider declaring a separate ”isInitialized” flag.

Client Comment *As this does not break anything, we see it as a future optimisation.*

```
61 if (address(universe) == address(0)) revert ZeroAddress(); // Check
      ↪ that the contract is initialised
```

CVF-9. INFO

- **Category** Unclear behavior
- **Source** savETHOriginGateway.sol

Description Precision loss is possible here.

Recommendation Consider either logging w raw, unscaled value, or explicitly checking that ”knotDETHBalanceInIndex” is a factor of 1 gwei.

Client Comment *Any beacon chain balance is GWEI denominated so there is no loss possible since the balance was previously scaled to WEI from beacon chain.*

```
83 knotDETHBalanceInIndex / 1 gwei,
```

CVF-10. INFO

- **Category** Unclear behavior
- **Source** savETHDestinationGateway.sol

Description In case the ”deposits” array is shorter than other arrays, the remaining elements of the other arrays are silently ignored.

Recommendation Consider explicitly checking that array lengths are the same.

Client Comment *Disputed since: batchPush does length checks.*

```
62 for (uint256 i; i < _deposits.length; ++i) {
```

CVF-11. INFO

- **Category** Suboptimal
- **Source** savETHDestinationGateway.sol

Description In case the ”_endorsements” array is longer than other arrays, the remaining elements of it are silently ignored.

Client Comment *We will add extra checks in future gateway release - this is not a mainnet commit.*

```
124 _endorsements[i]
```

CVF-12. INFO

- **Category** Suboptimal
- **Source** savETHDestinationGateway.sol

Description The expression ”domains[_depositMetadata.originChainId].” is calculated several times.

Recommendation Consider calculating once and reusing.

Client Comment *As this does not break anything, we see it as a future optimisation.*

```
141 if (address(domains[_depositMetadata.originChainId].accumulator) ==
       ↪ address(0)) revert ZeroAddress(); // Check that the gateway is
       ↪  configured
```

```
144 if (_depositMetadata.originGateway != domains[_depositMetadata.
       ↪ originChainId].destination) revert InvalidOrigin(); // ensure
       ↪ the origin gateway address is correct
```

```
202 domains[_depositMetadata.originChainId].accumulator.
       ↪ injectVectorCommitment(_balIncreaseDataRoot);
```

CVF-13. INFO

- **Category** Suboptimal
- **Source** savETHGateway.sol

Description This expression is calculated twice.

Recommendation Consider calculating once and reusing.

Client Comment *As this does not break anything, we see it as a future optimisation.*

```
82         universe.saveETHRegistry().associatedIndexIdForKnot(
              ↪ _baseParams.paramThree),
           _baseParams.paramThree
```

```
98 uint256 knotDETHBalanceInIndex = universe.saveETHRegistry().
       ↪ knotDETHBalanceInIndex(associatedIndexId, _baseParams.
       ↪ paramThree);
```

CVF-14. INFO

- **Category** Procedural
- **Source** StakeHouseUniverseAPI.sol

Description The name of this contract differs from the name of the file.

Recommendation Consider renaming either the contract of the file to make names the same.

Client Comment *As this does not break anything, we see it as a future optimisation.*

```
9 abstract contract StakeHouseUniverse {
```

CVF-15. INFO

- **Category** Documentation
- **Source** StakeHouseUniverseAPI.sol

Description It is unclear what flags are supported.

Recommendation Consider documenting.

Client Comment *Documented as part of Stakehouse.*

```
29 uint256 flags,          // Flags associated with the member
```

CVF-16. INFO

- **Category** Suboptimal
- **Source** RPBSVerificationLibrary.sol

Description The expression ”isUTXOEndorsedByPublicKey[_depositMetadata.depositLeafIndex]” is calculated multiple times.

Recommendation Consider calculating once before the loop.

Client Comment *As this does not break anything, we see it as a future optimisation.*

```
76 require(!isUTXOEndorsedByPublicKey[_depositMetadata.depositLeafIndex
   ↪ ][encodePointHex(_endorsements[i].publicKey)], "Already
   ↪ endorsed");
```

```
87 isUTXOEndorsedByPublicKey[_depositMetadata.depositLeafIndex][
   ↪ encodePointHex(_endorsements[i].publicKey)] = true;
```

CVF-17. FIXED

- **Category** Bad naming
- **Source** IRBSEndorserSignature.sol

Description The ”I” prefix usually marks interfaces, while here is it used for an abstract contract. Also, this abstract contract is defined in a file residing in the ”interfaces” directory.

Recommendation Consider either turning this abstract contract into an interface or removing the ”I” prefix and moving the file out of the ” interfaces” directory.

```
7 abstract contract IRBSEndorserSignature is RPBS {
```

CVF-18. FIXED

- **Category** Bad datatype
- **Source** IEIP721Signature.sol

Recommendation The name should be ”IEIP712Signature”.

```
6 interface IEIP721Signature {
```

CVF-19. INFO

- **Category** Bad naming
- **Source** IGatewayIngestor.sol

Description The semantics of the parameters is unclear.

Recommendation Consider giving descriptive names to the parameters and/or adding a documentation comment.

```
8 event Consume(address, uint256, bytes);
```

CVF-20. INFO

- **Category** Bad naming
- **Source** IGatewayIngestor.sol

Description The semantics of the arguments and the returned value is unclear.

Recommendation Consider giving descriptive names to the arguments and the returned value and/or adding a documentation comment.

```
10 function consume(address, uint256, bytes calldata) external returns
   ↪ (uint256);
```

8 Moderate Issues

CVF-21. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Description The function doesn't check whether recovery was already activalted for the domain, this it is possible to activate recovery several times.

Recommendation Consider preventing several recovery activations for the same domain.

Client Comment *We will add extra checks in future gateway release - this is not a mainnet commit.*

```
257 function triggerKillSwitch(uint256 _domainId, bool _killPush)
       ↪ external virtual onlyGatewayOperator {
```

CVF-22. INFO

- **Category** Overflow/Underflow
- **Source** savETHDestinationReporter.sol

Description Overflow is possible here.

Recommendation Consider using checked math.

Client Comment *There is only 120m supply of ether. uint256 is more than capable of storing this*

```
32 uint256 updatedActiveBalanceInWei = _totalDETHBalance * 1 gwei; //
       ↪ convert gwei to wei value
```

CVF-23. INFO

- **Category** Suboptimal
- **Source** Accumulator.sol

Description This implicitly forbids zero environment ID.

Recommendation Consider surrounding with ”unchecked” or forbidding explicitly.

Client Comment *Zero chain ID is not a posssibility so disagree.*

```
140 sha256(abi.encode(_environmentId - 1))
```

CVF-24. FIXED

- **Category** Suboptimal
- **Source** GatewayToken.sol

Description This function allows a minter to burn other people’s tokens with explicit approval. This is very dangerous.

Recommendation Consider either allow burning own tokens only, or consume approval from token owners.

Client Comment *We will simplify this token contract to ensure only gateway can mint and burn.*

```
64 function burn(address _recipient, uint256 _amount) external {
```

CVF-25. INFO

- **Category** Suboptimal
- **Source** savETHOriginGateway.sol

Description Precision loss is possible here.

Recommendation Consider either hashing a raw, unscaled value, or explicitly checking that ”knotDETHBalanceInIndex” is a factor of ”1 gwei”.

Client Comment *Beacon chain balances are denominated in GWEI so precision loss is not possible.*

```
139 knotDETHBalanceInIndex / 1 gwei,
```

CVF-26. INFO

- **Category** Flaw
- **Source** savETHDestinationGateway.sol

Recommendation There is no length check for this argument.

Client Comment *We will add extra checks in future gateway release - this is not a mainnet commit.*

```
112 RPBSEndorsement[][] calldata _endorsements
```

CVF-27. INFO

- **Category** Suboptimal
- **Source** savETHGateway.sol

Description This effectively rounds a balance down before hashing.

Recommendation Consider hashing the precise value.

Client Comment *Beacon chain balances are denominated in GWEI so precision loss is not possible.*

```
84 ) / 1 gwei,
```

9 Minor Issues

CVF-28. INFO

- **Category** Procedural
- **Source** Gateway.sol

Description Consider specifying as ”^0.8.0” unless there is something special about this particular version.

Recommendation Also relevant for: dETHDestinationIngestor.sol, EndorserRegistry.sol, savETHDestinationReporter.sol, Accumulator.sol, ERC20OriginIngestor.sol, ERC20DestinationIngestor.sol, dETHOriginIngestor.sol, GatewayToken.sol, ERC20OriginDispenser.sol, ERC20DestinationDispenser.sol, Recovery.sol, dETHOriginDispenser.sol, dETHDestinationDispenser.sol, ERC20Gateway.sol, savETHOriginGateway.sol, StakeHouseUniverseDestinationGateway.sol, savETHDestinationGateway.sol, savETHRegistryDestinationGateway.sol, savETHGateway.sol, StakeHouseUniverseAPI.sol, RPBSVerificationLibrary.sol, ConsentVerification.sol, AccumulatorFactory.sol, IRBSEndorserSignature.sol, IsavETHDispenser.sol, IAccountManager.sol, ISavETHRegistry.sol, ISavETHManager.sol, IGateway.sol, IEIP721Signature.sol, IGatewayOperatorControlCentre.sol, IGatewayDispenser.sol, IAccumulatorMetadata.sol, IGatewayIngestor.sol, IAccumulator.sol.

```
1 pragma solidity ^0.8.13;
```

CVF-29. INFO

- **Category** Procedural
- **Source** Gateway.sol

Description Declaring top-level errors in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the errors into the contract or moving them into a separate file.

```
23 error NotAGatewayOperator();
   error GatewayIsPaused();
   error DomainIsStillOperational();
   error KillSwitchEnabledForDomain();
   error DomainOperationsArePausedOrKilled();
   error InvalidIngestorInterface();
   error InvalidDispenserInterface();
30 error EmptyLeaf();
   error InvalidLeafIndex();
   error InvalidChain();
   error InvalidAccumulatorRoot();
   error InvalidProofSize();
   error OnlyGatewayAdmin();
   error ZeroUintValue();
   error LessThanBatch();
   error DestinationCannotBeCurrentDomain();
   error TotalExtendedNotZero();
40 error DomainAlreadyAdded();
   error RecoveryCheckpointNotInjected();
   error RecoveryRootAlreadyInjected();
   error RecoveryRootNotInjected();
   error UTXOSpent();
   error InvalidConsent();
   error InvalidConsentId();
   error InvalidRPBSEndorsement();
   error DestinationGatewayCannotBeThisGateway();
   error CannotRegisterMoreThanOneReturnPath();
```

CVF-30. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Recommendation These errors could be made more useful by adding some parameters to them.

```
23 error NotAGatewayOperator();
```

```
25 error DomainIsStillOperational();
   error KillSwitchEnabledForDomain();
   error DomainOperationsArePausedOrKilled();
   error InvalidIngestorInterface();
   error InvalidDispenserInterface();
30 error EmptyLeaf();
   error InvalidLeafIndex();
   error InvalidChain();
```

```
40 error DomainAlreadyAdded();
   error RecoveryCheckpointNotInjected();
   error RecoveryRootAlreadyInjected();
   error RecoveryRootNotInjected();
   error UTXOSpent();
   error InvalidConsent();
   error InvalidConsentId();
   error InvalidRPBSEndorsement();
   error DestinationGatewayCannotBeThisGateway();
   error CannotRegisterMoreThanOneReturnPath();
```

CVF-31. INFO

- **Category** Bad naming
- **Source** Gateway.sol

Description The word ”event” in an event name is redundant.

Recommendation Consider removing.

```
54 event DepositEvent(
```

CVF-32. INFO

- **Category** Procedural
- **Source** Gateway.sol

Recommendation Consider splitting the event in two if more fields need logging.

```
54 event DepositEvent(
```

CVF-33. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Description It is unlikely that the signature raw values actually need logging.

```
64 uint8 v,
   bytes32 r,
   bytes32 s
```

CVF-34. INFO

- **Category** Procedural
- **Source** Gateway.sol

Recommendation These three mappings could be merged into one whose keys are domain IDs and values are structs encapsulating the values of the original mappings.

```
121 mapping(uint256 => Domain) public domains;
```

```
124 mapping(uint256 => bool) public isPushKilled;
```

```
127 mapping(uint256 => bool) public isUTXOSpent;
```

CVF-35. INFO

- **Category** Procedural
- **Source** Gateway.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
170 constructor() initializer {}
```

CVF-36. INFO

- **Category** Bad datatype
- **Source** Gateway.sol

Recommendation The type of this argument should be ”ConsentVerification”.

```
176 address _consentVerificationModule,
```

CVF-37. INFO

- **Category** Bad datatype
- **Source** Gateway.sol

Recommendation The type of this argument should be ”RPBSVerificationLibrary”.

```
177 address _rpbsVerificationModule,
```

CVF-38. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Recommendation These checks are redundant as it is anyway possible to pass a dead module address.

```
184 if (_consentVerificationModule == address(0)) revert ZeroAddress();
    if (_rpbsVerificationModule == address(0)) revert ZeroAddress();
```

CVF-39. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Recommendation These checks are redundant as it is anyway possible to pass a dead address.

```
211 if (address(_domainMetadata.ingestionModule) == address(0)) revert
    ↪ ZeroAddress();
    if (address(_domainMetadata.dispenserModule) == address(0)) revert
    ↪ ZeroAddress();
    if (address(_domainMetadata.accumulator) == address(0)) revert
    ↪ ZeroAddress();
    if (address(_domainMetadata.destination) == address(0)) revert
    ↪ ZeroAddress();
```

CVF-40. INFO

- **Category** Unclear behavior
- **Source** Gateway.sol

Description This event is emitted even if nothing actually changed.

```
251 emit GatewayOperatorUpdated(_operator, _isEnabled);
```

CVF-41. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Recommendation This event should also have the ”_killPush” value as a parameter.

```
265 emit KillSwitchTriggered(_domainId);
```

CVF-42. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Recommendation This event should have the checkpoint information as a parameter.

```
280 emit DomainCheckpointInjected(_domainId);
```

CVF-43. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Recommendation This event should have the recovery Merkle root as a parameter.

```
297 emit DomainRecoveryMerkleRootInjected(_domainId);
```

CVF-44. INFO

- **Category** Unclear behavior
- **Source** Gateway.sol

Description These events are emitted even if nothing actually changed.

```
303 emit GatewayPaused();
```

```
309 emit GatewayUnpaused();
```

CVF-45. INFO

- **Category** Unclear behavior
- **Source** Gateway.sol

Description This event is emitted even if nothing actually changed.

```
321 emit DomainPaused(_domainId);
```

CVF-46. INFO

- **Category** Unclear behavior
- **Source** Gateway.sol

Description This event is emitted even if nothing actually changed.

```
332 emit DomainUnPaused(_domainId);
```

CVF-47. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Recommendation It would be more efficient to pass a single array of structs with three fields instead of three parallel arrays. This would also make the length checks unnecessary.

```
340 bytes32[] calldata _transactionSummaries,
    uint256[] calldata _domainIds,
    BaseInputParams[] calldata _baseParams
```

CVF-48. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Recommendation It would be more efficient to pass a single array of structs with four fields instead of four parallel arrays. This would also make the length checks unnecessary.

```
376 bytes32[] calldata _transactionSummaries,
    DepositMetadata[] calldata _deposits,
    bytes32[][] calldata _accumulatorProofs,
    RPBSEndorsement[][] calldata _endorsements
```

CVF-49. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Description The ”verify” function never returns false.

Recommendation Consider removing the conditional statement.

```
416 if (!rpbsVerificationModule.verify(
```

CVF-50. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Recommendation The ”TREE_DEPTH” constant should be used here instead of a hard-coded value.

```
451 if (_depositCount > (2**32 - 1)) revert InvalidLeafIndex();
```

```
454 if (_proof.length != 32) revert InvalidProofSize();
```

```
460 for (uint256 i; i < 32; i++) {
```

CVF-51. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Recommendation Shift and bitwise AND instead of division and remainder operators, would be more efficient here,

```
461 if ((_index / 2 ** i) % 2 != 0) {
```

CVF-52. INFO

- **Category** Suboptimal
- **Source** Gateway.sol

Description This is needed if ‘zero_hashes‘ do not have preimages, so that different depositCounts would result in different roots.

```
468 // every merkle root is wrapped with the total deposit count at time
    ↪  of transaction and padding in order to form the accumulator
    ↪ root
node = sha256(abi.encodePacked(
470         node,
            _depositCount,
            getDomainContextForAccumulatorRoot(_originEnvironmentId)
    ));
```

CVF-53. INFO

- **Category** Procedural
- **Source** Gateway.sol

Description The expression ”domains[_depositMetadata.originChainId].dispenserModule” is calculated several times.

Recommendation Consider calculating once and reusing.

```
629 if (domains[_depositMetadata.originChainId].dispenserModule.
       ↪ isRecoveryEnabled()) {

631         domains[_depositMetadata.originChainId].dispenserModule.
               ↪ destinationRecoveryCheckpoint() == 0

635         domains[_depositMetadata.originChainId].dispenserModule.
               ↪ recoveryMerkleRoot() == bytes32(0)

638     totalProcessed = domains[_depositMetadata.originChainId].
           ↪ dispenserModule.dispenseViaRecovery(

664     totalProcessed = domains[_depositMetadata.originChainId].
           ↪ dispenserModule.dispense(
```

CVF-54. INFO

- **Category** Procedural
- **Source** Gateway.sol

Description The expression ”domains[_depositMetadata.originChainId]” is calculated several times.

Recommendation Consider calculating once and reusing.

```
629 if (domains[_depositMetadata.originChainId].dispenserModule.
       ↪ isRecoveryEnabled()) {
631         domains[_depositMetadata.originChainId].dispenserModule.
               ↪ destinationRecoveryCheckpoint() == 0
635         domains[_depositMetadata.originChainId].dispenserModule.
               ↪ recoveryMerkleRoot() == bytes32(0)
638     totalProcessed = domains[_depositMetadata.originChainId].
           ↪ dispenserModule.dispenseViaRecovery(
664     totalProcessed = domains[_depositMetadata.originChainId].
           ↪ dispenserModule.dispense(
675     domains[_depositMetadata.originChainId].accumulator.
           ↪ injectVectorCommitment(_transactionSummary);
681     domains[_depositMetadata.originChainId].totalExtended +=
           ↪ totalProcessed;
683     domains[_depositMetadata.originChainId].totalExtended -=
           ↪ totalProcessed;
```

CVF-55. INFO

- **Category** Procedural
- **Source** Gateway.sol

Recommendation These braces are redundant.

```
686 {
696 }
```

CVF-56. INFO

- **Category** Bad datatype
- **Source** dETHDestinationIngestor.sol

Recommendation The type of this variable should be more specific.

```
25 address public gateway;
```

CVF-57. INFO

- **Category** Procedural
- **Source** dETHDestinationIngestor.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
28 constructor() initializer {}
```

CVF-58. INFO

- **Category** Bad datatype
- **Source** dETHDestinationIngestor.sol

Recommendation The type of this argument should be ”StakeHouseUniverseDestinationGateway”.

```
31 address _universe,
```

CVF-59. INFO

- **Category** Bad datatype
- **Source** dETHDestinationIngestor.sol

Recommendation The type of this argument should be more specific.

```
32 address _gateway
```

CVF-60. INFO

- **Category** Suboptimal
- **Source** dETHDestinationIngestor.sol

Recommendation These checks are redundant as it is anyway possible to pass dead addresses.

```
34 if (_universe == address(0)) revert ZeroAddress();
   if (_gateway == address(0)) revert ZeroAddress();
```

CVF-61. INFO

- **Category** Bad naming
- **Source** dETHDestinationIngestor.sol

Recommendation The event names can be more specific.

```
34 if (_universe == address(0)) revert ZeroAddress();
   if (_gateway == address(0)) revert ZeroAddress();
```

CVF-62. INFO

- **Category** Bad datatype
- **Source** dETHDestinationIngestor.sol

Recommendation The type of the "_stakeHouse" argument should be more specific.

```
45 function consume(address _stakeHouse, uint256 _paramTwo, bytes
     ↪ calldata _blsPubKey) external override returns (uint256) {
```

CVF-63. INFO

- **Category** Bad naming
- **Source** dETHDestinationIngestor.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or describing in a documentation comment.

```
45 function consume(address _stakeHouse, uint256 _paramTwo, bytes
   ↪ calldata _blsPubKey) external override returns (uint256) {
```

CVF-64. INFO

- **Category** Bad naming
- **Source** EndorserRegistry.sol

Recommendation Events are usually named via nouns, such as ”Endorser” and ”EndorserLifecycleStatus”.

```
13 event EndorserRegistered(string RPBSPublicKey);
```

```
16 event EndorserLifecycleStatusUpdated(string RPBSPublicKey);
```

CVF-65. INFO

- **Category** Unclear behavior
- **Source** EndorserRegistry.sol

Description These event should have some kind of endorser ID as indexed parameters.

```
13 event EndorserRegistered(string RPBSPublicKey);
```

```
16 event EndorserLifecycleStatusUpdated(string RPBSPublicKey);
```

CVF-66. INFO

- **Category** Suboptimal
- **Source** EndorserRegistry.sol

Recommendation This event should have the updated status as a parameter.

```
16 event EndorserLifecycleStatusUpdated(string RPBSPublicKey);
```

CVF-67. INFO

- **Category** Procedural
- **Source** EndorserRegistry.sol

Recommendation These two mappings could be merged into one by introducing a new endorser lifecycle status: ”UNREGISTERED”.

```
27 mapping(string => EndorserLifecycleStatus) public
      ↪ lifecycleStatusOfEndorserByPublicKey;
```

```
30 mapping(string => bool) public isEndorserRegistered;
```

CVF-68. INFO

- **Category** Procedural
- **Source** EndorserRegistry.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
36 constructor() initializer {}
```

CVF-69. INFO

- **Category** Suboptimal
- **Source** EndorserRegistry.sol

Recommendation This check is redundant as it is anyway possible to pass a dead owner address.

```
39 require(_owner != address(0), "Owner is zero address");
```

CVF-70. INFO

- **Category** Suboptimal
- **Source** EndorserRegistry.sol

Recommendation The first part of this check is redundant, as it is not possible to set ”ACTIVE” status for an unregistered endorser.

```
86 return isEndorserRegistered[pubKey] &&
          lifecycleStatusOfEndorserByPublicKey[pubKey] ==
             ↪ EndorserLifecycleStatus.ACTIVE;
```

CVF-71. INFO

- **Category** Procedural
- **Source** savETHDestinationReporter.sol

Description We didn’t review these files.

```
5 import { StakeHouseUniverse } from "../../../StakeHouseUniverse.sol"
     ↪ ;
  import { savETHRegistry } from "../../../banking/savETHRegistry.sol"
     ↪ ;
  import { StakeHouseUUPSCoreModule } from "../../../proxies/
     ↪ StakeHouseUUPSCoreModule.sol";
```

CVF-72. INFO

- **Category** Procedural
- **Source** savETHDestinationReporter.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
13 constructor() initializer {}
```

CVF-73. INFO

- **Category** Bad datatype
- **Source** savETHDestinationReporter.sol

Recommendation The type of this argument should be more specific.

```
24 address _stakeHouse,
```

CVF-74. INFO

- **Category** Procedural
- **Source** savETHDestinationReporter.sol

Description 256 bits is enough to represent a raw, unscaled value. Scaled values just make code more complicated, less efficient and more error-prone.

Recommendation Consider using raw, unscaled values.

```
26 uint256 _totalDETHBalance // gwei value
```

CVF-75. INFO

- **Category** Suboptimal
- **Source** savETHDestinationReporter.sol

Recommendation The values ”24 ether” and ”1 gwei” should be named constants.

```
31 uint256 highestSeenBalance = 24 ether + saveETHRegistry.
      ↪ dETHRewardsMintedForKnot(_memberId); // a wei value
   uint256 updatedActiveBalanceInWei = _totalDETHBalance * 1 gwei; //
      ↪ convert gwei to wei value
```

CVF-76. INFO

- **Category** Suboptimal
- **Source** Accumulator.sol

Description Declaring top-level errors in a file named after a contracts makes it harder to navigate through code.

Recommendation Consider moving the errors into the contract.

```
12 error TreeFull();
   error InvalidManager();
   error DepositAlreadyPushed();
```

CVF-77. INFO

- **Category** Suboptimal
- **Source** Accumulator.sol

Recommendation These error could be made more useful by adding some parameters to them.

```
13 error InvalidManager();
   error DepositAlreadyPushed();
```

CVF-78. INFO

- **Category** Suboptimal
- **Source** Accumulator.sol

Recommendation These two variables could be replaced with a single variable of type ”uint256[]”.

```
29 uint256 public override depositCount;
```

```
41 mapping(uint256 => uint256) public indexedHashStamp;
```

CVF-79. INFO

- **Category** Suboptimal
- **Source** Accumulator.sol

Recommendation This variable should be declared as immutable.

```
38 uint256 public originId;
```

CVF-80. INFO

- **Category** Documentation
- **Source** Accumulator.sol

Description The semantics of keys and values in this mapping is unclear.

Recommendation Consider documenting.

```
41 mapping(uint256 => uint256) public indexedHashStamp;
```

CVF-81. INFO

- **Category** Procedural
- **Source** Accumulator.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
47 constructor() initializer {}
```

CVF-82. INFO

- **Category** Suboptimal
- **Source** Accumulator.sol

Description This check is redundant as it is anyway possible to pass a dead accumulator manager address.

Recommendation Consider removing this check.

```
51 if (_accumulatorManager == address(0)) revert ZeroAddress();
```

CVF-83. INFO

- **Category** Suboptimal
- **Source** Accumulator.sol

Description Here the previous zero hash just written into the storage is read back.

Recommendation Consider reusing the written value.

```
60 zero_hashes[height + 1] = sha256(abi.encodePacked(zero_hashes[height
   ↪ ], zero_hashes[height]));
```

CVF-84. INFO

- **Category** Procedural
- **Source** Accumulator.sol

Recommendation The check should be performed earlier, before modifying the state.

```
73 if (depositCount == MAX_DEPOSIT_COUNT) revert TreeFull();
```

CVF-85. INFO

- **Category** Procedural
- **Source** Accumulator.sol

Description The value of the ”depositCount” variable is read several times.

Recommendation Consider reading once and reusing.

```
73 if (depositCount == MAX_DEPOSIT_COUNT) revert TreeFull();
```

```
77 indexedHashStamp[depositCount] = block.timestamp;
```

```
81     depositCount += 1;
```

CVF-86. INFO

- **Category** Suboptimal
- **Source** Accumulator.sol

Description The ”depositCount” value just written to the storage is read back.

Recommendation Consider reusing the written value.

```
81 depositCount += 1;
   uint256 size = depositCount;
```

CVF-87. INFO

- **Category** Suboptimal
- **Source** Accumulator.sol

Recommendation It would be more efficient to just remove the loop condition to make the loop effectively ”infinite”.

```
100 // this code should be unreachable. We assert `false` just to be
       ↪ safe.
    assert(false);
```

CVF-88. INFO

- **Category** Procedural
- **Source** ERC20OriginIngestor.sol

Description Declaring top-level errors in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the errors into the contract or moving them into a separate file.

```
12 error OnlyGateway();
```

CVF-89. INFO

- **Category** Procedural
- **Source** ERC20OriginIngestor.sol

Recommendation This interface should be moved into a separate file named ”IERC20Gateway.sol”.

```
14 interface IERC20Gateway {
```

CVF-90. INFO

- **Category** Bad datatype
- **Source** ERC20OriginIngestor.sol

Recommendation The return type should be ”IERC20”.

```
15 function token() external view returns (address);
```

CVF-91. INFO

- **Category** Bad datatype
- **Source** ERC20OriginIngestor.sol

Recommendation The type of this variable should be more specific.

```
23 address public dispenser;
```

CVF-92. INFO

- **Category** Bad datatype
- **Source** ERC20OriginIngestor.sol

Recommendation The type of this variable should be ”IERC20Gateway”.

```
26 address public gateway;
```

CVF-93. INFO

- **Category** Procedural
- **Source** ERC20OriginIngestor.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
29 constructor() initializer {}
```

CVF-94. INFO

- **Category** Bad datatype
- **Source** ERC20OriginIngestor.sol

Recommendation The type of this argument should be more specific.

```
32 address _dispenser,
```

CVF-95. INFO

- **Category** Bad datatype
- **Source** ERC20OriginIngestor.sol

Recommendation The type of this argument should be ”IERC20Gateway”.

```
33 address _gateway
```

CVF-96. INFO

- **Category** Suboptimal
- **Source** ERC20OriginIngestor.sol

Recommendation These checks are redundant as it is anyway possible to pass dead addresses.

```
35 if (_dispenser == address(0)) revert ZeroAddress();
   if (_gateway == address(0)) revert ZeroAddress();
```

CVF-97. INFO

- **Category** Bad naming
- **Source** ERC20OriginIngestor.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or describing in a documentation comment.

```
46 ) external override returns (uint256) {
```

CVF-98. INFO

- **Category** Procedural
- **Source** ERC20DestinationIngestor.sol

Description Declaring top-level error in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the errors into the contract or moving them into a separate file.

```
11 error OnlyGateway();
```

CVF-99. INFO

- **Category** Procedural
- **Source** ERC20DestinationIngestor.sol

Recommendation This interface should be moved into a separate file named ”IERC20Gateway.sol”.

```
13 interface IERC20Gateway {
```

CVF-100. INFO

- **Category** Bad datatype
- **Source** ERC20DestinationIngestor.sol

Recommendation The return type should be ”GatewayToken”.

```
14 function token() external view returns (address);
```

CVF-101. INFO

- **Category** Bad datatype
- **Source** ERC20DestinationIngestor.sol

Recommendation The type of this variable should be ”IERC20Gateway”.

```
20 address public gateway;
```

CVF-102. INFO

- **Category** Procedural
- **Source** ERC20DestinationIngestor.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
23 constructor() initializer {}
```

CVF-103. INFO

- **Category** Suboptimal
- **Source** ERC20DestinationIngestor.sol

Recommendation This check is redundant as it is anyway possible to pass a dead gateway address.

```
26 if (_gateway == address(0)) revert ZeroAddress();
```

CVF-104. INFO

- **Category** Bad naming
- **Source** ERC20DestinationIngestor.sol

Description The semantics of the returned value is unclear.

Recommendation Consider either giving a descriptive name to the returned value and/or describing in a documentation comment.

```
35 ) external override returns (uint256) {
```

CVF-105. INFO

- **Category** Procedural
- **Source** dETHOriginIngestor.sol

Description Declaring top-level errors in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the errors into the contract or moving them into a separate file.

```
16 error OnlyGateway();
   error InactiveKnot();
   error KnotKickedFromBeaconChain();
   error KnotHasExited();
```

CVF-106. INFO

- **Category** Bad datatype
- **Source** dETHOriginIngestor.sol

Recommendation The type of this argument should be more specific.

```
40 address _gateway,
```

CVF-107. INFO

- **Category** Bad datatype
- **Source** dETHOriginIngestor.sol

Recommendation The type of this argument should be ”IsavETHDispenser”.

```
41 address _dispenser,
```

CVF-108. INFO

- **Category** Bad datatype
- **Source** dETHOriginIngestor.sol

Recommendation The type of this argument should be ”StakeHouseUniverse”.

```
42 address _universe
```

CVF-109. INFO

- **Category** Suboptimal
- **Source** dETHOriginIngestor.sol

Recommendation These checks are redundant as it is anyway possible to pass dead addresses.

```
44 if (_gateway == address(0)) revert ZeroAddress();
   if (_dispenser == address(0)) revert ZeroAddress();
   if (_universe == address(0)) revert ZeroAddress();
```

CVF-110. INFO

- **Category** Bad datatype
- **Source** dETHOriginIngestor.sol

Recommendation The type of this argument should be more specific.

```
55 address _stakeHouse,
```

CVF-111. INFO

- **Category** Bad naming
- **Source** dETHOriginIngestor.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or describing in a documentation comment.

```
58 ) external override returns (uint256) {
```

CVF-112. INFO

- **Category** Procedural
- **Source** GatewayToken.sol

Description Declaring top-level errors in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the errors into the contract or moving them into a separate file.

```
9  error OnlyAdmin();
10 error NotAuthorized();
```

CVF-113. INFO

- **Category** Bad naming
- **Source** GatewayToken.sol

Recommendation Events are usually named via nouns, such as ”Admin” or ”Minter”.

```
16 event AdminUpdated(address indexed admin, bool isEnabled);
```

```
19 event MinterUpdated(address indexed minter, bool isEnabled);
```

CVF-114. INFO

- **Category** Procedural
- **Source** GatewayToken.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
28 constructor() initializer {}
```

CVF-115. INFO

- **Category** Unclear behavior
- **Source** GatewayToken.sol

Description These events are emitted even if nothing actually changed.

```
47 emit MinterUpdated(_minter, _isEnabled);
```

```
54 emit AdminUpdated(_admin, _isEnabled);
```

CVF-116. INFO

- **Category** Procedural
- **Source** ERC20OriginDispenser.sol

Recommendation This interface should be moved into a separate file named ”IERC20Gateway.sol”.

```
13 interface IERC20Gateway {
```

CVF-117. INFO

- **Category** Bad datatype
- **Source** ERC20OriginDispenser.sol

Recommendation The return type should be ”IERC20”.

```
14 function token() external view returns (address);
```

CVF-118. INFO

- **Category** Procedural
- **Source** ERC20OriginDispenser.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
22 constructor() initializer {}
```

CVF-119. INFO

- **Category** Bad datatype
- **Source** ERC20OriginDispenser.sol

Recommendation The argument type should be ”IERC20Gateway”.

```
24 function init(address _gateway) external initializer {
```

CVF-120. INFO

- **Category** Bad naming
- **Source** ERC20OriginDispenser.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or describing in a documentation comment.

```
29 function _performDispense(uint256 _amount, address _recipient,
   ↪ uint256, bytes calldata _data) internal override returns (
   ↪ uint256) {
```

CVF-121. INFO

- **Category** Procedural
- **Source** ERC20DestinationDispenser.sol

Recommendation This interface should be moved into a separate file named ”IERC20Gateway.sol”.

```
8 interface IERC20Gateway {
```

CVF-122. INFO

- **Category** Bad datatype
- **Source** ERC20DestinationDispenser.sol

Recommendation The return type should be ”GatewayToken”.

```
9 function token() external view returns (address);
```

CVF-123. INFO

- **Category** Procedural
- **Source** ERC20DestinationDispenser.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
15 constructor() initializer {}
```

CVF-124. INFO

- **Category** Bad datatype
- **Source** ERC20DestinationDispenser.sol

Recommendation The argument type should be more ”IERC20Gateway”.

```
17 function init(address _gateway) external initializer {
```

CVF-125. INFO

- **Category** Bad naming
- **Source** ERC20DestinationDispenser.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or describing in a documentation comment.

```
22 function _performDispense(uint256 _amount, address _recipient,
      ↪ uint256, bytes calldata _data) internal override returns (
      ↪ uint256) {
```

CVF-126. INFO

- **Category** Procedural
- **Source** Recovery.sol

Description Declaring top-level errors in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the errors into the contract or moving them into a separate file.

```
13 error DispenseViaRecoveryEnabled();
   error RecoveryIsNotEnabled();
   error NotPartOfMerkleTree();
   error AlreadyRecovered();
   error OnlyGateway();
   error MerkleRootAlreadyInjected();
   error MerkleRootNotInjected();
20 error ZeroMerkleRoot();
   error DestinationCheckpointAlreadySet();
   error UintIsZero();
   error DestinationCheckpointIsNotSet();
```

CVF-127. INFO

- **Category** Bad naming
- **Source** Recovery.sol

Description The error name is too generic.

Recommendation Consider making the name more specific, such as ”DestinationRecoveryCheckpointIsZero”.

```
22 error UintIsZero();
```

CVF-128. INFO

- **Category** Bad naming
- **Source** Recovery.sol

Recommendation Events are usually named via nouns, such as ”Recovery”, ”RecoveryMerkleRoot”, etc.

```
29 event RecoveryActivated();
```

```
32 event RecoveryMerkleRootInjected();
```

```
35 event DestinationRecoveryCheckpointFinalized();
```

CVF-129. INFO

- **Category** Bad datatype
- **Source** Recovery.sol

Recommendation The type of this variable should be more specific.

```
44 address public gateway;
```

CVF-130. INFO

- **Category** Bad datatype
- **Source** Recovery.sol

Recommendation The argument type should be more specific.

```
55 function __Recovery_init(address _gateway) internal {
```

CVF-131. INFO

- **Category** Bad naming
- **Source** Recovery.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or describing in a documentation comment.

```
103 ) external virtual override returns (uint256) {
```

```
124 ) external override virtual returns (uint256) {
```

```
159 function _performDispense(uint256, address, uint256, bytes calldata)
    ↪  internal virtual returns (uint256);
```

CVF-132. INFO

- **Category** Procedural
- **Source** dETHOriginDispenser.sol

Description We didn't review this file.

```
9 } from "../../../gateway-implementations/dETH/StakeHouseUniverseAPI.
    ↪ sol";
```

CVF-133. INFO

- **Category** Procedural
- **Source** dETHOriginDispenser.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
30 constructor() initializer {}
```

CVF-134. INFO

- **Category** Bad datatype
- **Source** dETHOriginDispenser.sol

Recommendation The type of this argument should be ”StakeHouseUniverse”.

```
33 address _universe,
```

CVF-135. INFO

- **Category** Bad datatype
- **Source** dETHOriginDispenser.sol

Recommendation The type of this argument should be more specific.

```
34 address _gateway
```

CVF-136. INFO

- **Category** Suboptimal
- **Source** dETHOriginDispenser.sol

Recommendation These checked are redundant as it is anyway possible to pass dead addresses.

```
36 if (_universe == address(0)) revert ZeroAddress();
   if (_gateway == address(0)) revert ZeroAddress();
```

CVF-137. INFO

- **Category** Bad datatype
- **Source** dETHOriginDispenser.sol

Recommendation The type of this argument should be more specific.

```
51 address _stakeHouse,
```

CVF-138. INFO

- **Category** Bad naming
- **Source** dETHOriginDispenser.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or adding a documentation comment.

```
54 ) internal override returns (uint256) {
```

CVF-139. INFO

- **Category** Suboptimal
- **Source** dETHOriginDispenser.sol

Recommendation The ”1 gwei” value should be a named constant.

```
61 (uint256(_amount) * 1 gwei) > universe.savETHMan().
     ↪ knotDETHBalanceInIndex(gatewayIndex, _blsPubKey)
```

CVF-140. INFO

- **Category** Procedural
- **Source** dETHDestinationDispenser.sol

Description Declaring top-level errors in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the errors into the contract or moving them into a separate file.

```
13 error InvalidState();
```

CVF-141. INFO

- **Category** Procedural
- **Source** dETHDestinationDispenser.sol

Recommendation This interface should be moved into a separate file named ”dETHIngestor.sol”.

```
15 interface dETHIngestor {
```

CVF-142. INFO

- **Category** Bad datatype
- **Source** dETHDestinationDispenser.sol

Recommendation The type of this variable should be more specific.

```
28 address public ingestor;
```

CVF-143. INFO

- **Category** Procedural
- **Source** dETHDestinationDispenser.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
31 constructor() initializer {}
```

CVF-144. INFO

- **Category** Bad datatype
- **Source** dETHDestinationDispenser.sol

Recommendation The type of this argument should be ”StakeHouseUniverseDestinationGateway”.

```
34 address _universe,
```

CVF-145. INFO

- **Category** Bad datatype
- **Source** dETHDestinationDispenser.sol

Recommendation The type of these arguments should be more specific.

```
35 address _gateway,
   address _ingestor
```

CVF-146. INFO

- **Category** Suboptimal
- **Source** dETHDestinationDispenser.sol

Recommendation These checks are redundant as it is anyway possible to pass dead addresses.

```
38 if (_universe == address(0)) revert ZeroAddress();
   if (_gateway == address(0)) revert ZeroAddress();
40 if (_ingestor == address(0)) revert ZeroAddress();
```

CVF-147. INFO

- **Category** Bad datatype
- **Source** dETHDestinationDispenser.sol

Recommendation The type of this argument should be more specific.

```
56 address _stakeHouse,
```

CVF-148. INFO

- **Category** Bad naming
- **Source** dETHDestinationDispenser.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or adding a documentation comment.

```
59 ) internal override returns (uint256) {
```

CVF-149. INFO

- **Category** Bad datatype
- **Source** ERC20Gateway.sol

Recommendation The type of this argument should be ”IERC20”.

```
22 address _token,
```

CVF-150. INFO

- **Category** Bad datatype
- **Source** ERC20Gateway.sol

Recommendation The type of this argument should be ”ConsentVerification”.

```
25 address _consentVerificationModule,
```

CVF-151. INFO

- **Category** Bad datatype
- **Source** ERC20Gateway.sol

Recommendation The type of this argument should be ”IRPBSVerificationLibrary”.

```
26 address _rpbsVerificationModule,
```

CVF-152. INFO

- **Category** Suboptimal
- **Source** ERC20Gateway.sol

Recommendation This check is redundant as it is anyway possible to pass a dead token address.

```
30 if (_token == address(0)) revert ZeroAddress();
```

CVF-153. INFO

- **Category** Procedural
- **Source** savETHOriginGateway.sol

Description Declaring top-level errors in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the errors into the contract or moving them into a separate file.

```
17 error UnknownAsset();
   error InactiveKnot();
```

CVF-154. INFO

- **Category** Bad naming
- **Source** savETHOriginGateway.sol

Recommendation Events are usually named via nouns, such as ”Poke”.

```
23 event Poked(
```

CVF-155. INFO

- **Category** Bad datatype
- **Source** savETHOriginGateway.sol

Recommendation The type of this parameter should be more specific.

```
25 address indexed house,          // Associated stakehouse for BLS
      ↪ public key
```

CVF-156. INFO

- **Category** Suboptimal
- **Source** savETHOriginGateway.sol

Recommendation It would be more efficient to pass a single array of structs with three fields, rather than three parallel arrays. This would also make the length checks unnecessary.

```
38 address[] calldata _stakeHouses,
   bytes[] calldata _blsPublicKeys,
40 bytes32[] calldata _pokeDataRoots
```

CVF-157. INFO

- **Category** Procedural
- **Source** savETHOriginGateway.sol

Description The expression ”domains[_domainId].” is calculated several times.

Recommendation Consider calculating once and reusing.

```
68 uint256 gatewayIndex = IsavETHDispenser(address(domains[_domainId].
      ↪ dispenserModule)).gatewayIndex();
```

```
84     domains[_domainId].accumulator.depositCount(),
```

```
93 domains[_domainId].accumulator.injectVectorCommitment(_pokeDataRoot)
      ↪ ;
```

CVF-158. INFO

- **Category** Bad datatype
- **Source** savETHOriginGateway.sol

Recommendation The values "1 gwei" and "24 ether" should be named constants.

```
83     knotDETHBalanceInIndex / 1 gwei,
```

```
119 if (knotDETHBalanceInIndex <= 24 ether) revert InvalidKnotBalance();
```

```
126 if (knotDETHBalanceInIndex / 1 gwei > type(uint64).max) revert
    ↪ DepositTooHigh();
```

```
139     knotDETHBalanceInIndex / 1 gwei,
```

CVF-159. INFO

- **Category** Procedural
- **Source** savETHOriginGateway.sol

Recommendation This check should be performed before emitting an event.

```
89  if (
90      _getPokeLeaf(_domainId, _stakeHouse, _blsPublicKey,
        ↪ knotDETHBalanceInIndex) != _pokeDataRoot
    ) revert InvalidSummary();
```

CVF-160. INFO

- **Category** Suboptimal
- **Source** savETHOriginGateway.sol

Recommendation It would be cheaper to check "knotDETHBalanceInIndex > type(uint64).max * 1 gwei" as the right part could be computed at compile time.

```
126 if (knotDETHBalanceInIndex / 1 gwei > type(uint64).max) revert
    ↪ DepositTooHigh();
```

CVF-161. INFO

- **Category** Procedural
- **Source** StakeHouseUniverseDestinationGateway.sol

Description We didn't review these files.

```
7  import { StakeHouseAccessControls } from '../../../
       ↪ StakeHouseAccessControls.sol';
   import { StakeHouseUniverse } from '../../../StakeHouseUniverse.sol'
       ↪ ;
```

```
11 import { StakeHouseUUPSCoreModule } from "../../../proxies/
       ↪ StakeHouseUUPSCoreModule.sol";
```

CVF-162. INFO

- **Category** Procedural
- **Source** StakeHouseUniverseDestinationGateway.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
30 constructor() initializer {}
```

CVF-163. INFO

- **Category** Bad datatype
- **Source** StakeHouseUniverseDestinationGateway.sol

Recommendation The type of this argument should be "savETHRegistryDestinationGateway".

```
35 address _saveETHRegistryLogic,
```

CVF-164. INFO

- **Category** Bad datatype
- **Source** StakeHouseUniverseDestinationGateway.sol

Recommendation The type of these arguments should be more specific.

```
36 address _dETHLogic,
   address _saveETHLogic,
```

CVF-165. INFO

- **Category** Bad datatype
- **Source** StakeHouseUniverseDestinationGateway.sol

Recommendation The type of this argument should be ”savETHDestinationReporter”.

```
38 address _savETHManagerLogic,
```

CVF-166. INFO

- **Category** Bad datatype
- **Source** StakeHouseUniverseDestinationGateway.sol

Recommendation The type of this argument should be ”savETHDestinationReporter”.

```
39 address _reporterLogic
```

CVF-167. INFO

- **Category** Bad datatype
- **Source** StakeHouseUniverseDestinationGateway.sol

Recommendation The return type should be more specific.

```
78 function memberKnotToStakeHouse(bytes calldata _blsPubKey) external
      ↪ view returns (address) {
```

CVF-168. INFO

- **Category** Procedural
- **Source** savETHDestinationGateway.sol

Description Defining top-level errors in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the errors into the contract or moving then into a separate file.

```
19 error UnknownAsset();
20 error NoBalanceIncreaseDetected();
   error InvalidRoot();
   error InvalidRPBSEndorsement();
```

CVF-169. INFO

- **Category** Bad naming
- **Source** savETHDestinationGateway.sol

Recommendation Events are usually named via nouns, such as ”Poke”.

```
32 event BalancePokedSuccessfully(
```

CVF-170. INFO

- **Category** Bad datatype
- **Source** savETHDestinationGateway.sol

Recommendation The type of these parameters should be more specific.

```
34 address originGateway,              // Address of gateway that
      ↪ processed the poke
   address indexed house,              // Associated Stakehouse registry
      ↪  for bls public key
```

CVF-171. INFO

- **Category** Suboptimal
- **Source** savETHDestinationGateway.sol

Recommendation It would be more efficient to pass a single array of structs with fours fields, rather then four parallel arrays. This would also make the length checks unnecessary.

```
49 bytes32[] calldata _transactionSummaries,
50 DepositMetadata[] calldata _deposits,
   bytes32[][] calldata _accumulatorProofs,
   RPBSEndorsement[][] calldata _endorsements
```

CVF-172. INFO

- **Category** Suboptimal
- **Source** savETHDestinationGateway.sol

Recommendation It would be more efficient to pass a single array of structs with two fields, rather than two parallel arrays. This would also make the length check unnecessary.

```
83 bytes32[] calldata _originAccumulatorProof,
   RPBSEndorsement[] calldata _endorsements
```

CVF-173. INFO

- **Category** Suboptimal
- **Source** savETHDestinationGateway.sol

Recommendation It would be more efficient to pass a single array of structs with fours fields, rather then four parallel arrays. This would also make the length checks unnecessary.

```
109 bytes32[] calldata _balIncreaseDataRoots,
110 DepositMetadata[] calldata _deposits,
    bytes32[][] calldata _originAccumulatorProofs,
    RPBSEndorsement[][] calldata _endorsements
```

CVF-174. INFO

- **Category** Suboptimal
- **Source** savETHDestinationGateway.sol

Recommendation It would be more efficient to pass a single array of structs with fours fields, rather then four parallel arrays. This would also make the length checks unnecessary.

138
```
bytes32[] calldata _originAccumulatorProof,
RPBSEndorsement[] calldata _endorsements
```

CVF-175. INFO

- **Category** Procedural
- **Source** savETHDestinationGateway.sol

Description The expression ”domains[_depositMetadata.originChainId].accumulator” is calculated twice.

Recommendation Consider calculating once and reusing.

141
```
if (address(domains[_depositMetadata.originChainId].accumulator) ==
   ↪ address(0)) revert ZeroAddress(); // Check that the gateway is
   ↪  configured
```

202
```
domains[_depositMetadata.originChainId].accumulator.
   ↪ injectVectorCommitment(_balIncreaseDataRoot);
```

CVF-176. INFO

- **Category** Bad datatype
- **Source** savETHDestinationGateway.sol

Recommendation The values ”24 ether” and ”1 gwei” should be named constant.

159
```
if (_depositMetadata.amount <= (24 ether / 1 gwei)) revert
   ↪ NoBalanceIncreaseDetected();
```

CVF-177. INFO

- **Category** Procedural
- **Source** savETHRegistryDestinationGateway.sol

Description We didn't review this file.

```
5 import { savETHRegistry } from "../../../banking/savETHRegistry.sol"
    ↪ ;
```

CVF-178. INFO

- **Category** Bad datatype
- **Source** savETHRegistryDestinationGateway.sol

Recommendation The key type for this mapping should be more specific.

```
11 mapping(address => bool) public isValidHouse;
```

CVF-179. INFO

- **Category** Bad datatype
- **Source** savETHRegistryDestinationGateway.sol

Recommendation The value type for this mapping should be more specific.

```
14 mapping(bytes => address) public associatedHouseForKnot;
```

CVF-180. INFO

- **Category** Bad datatype
- **Source** savETHRegistryDestinationGateway.sol

Recommendation The type of the "_stakeHouse" arguments should be more specific.

```
17 function setIsValidHouse(address _stakeHouse) external onlyModule {
```

```
22 function setAssociatedHouseForKnot(address _stakeHouse, bytes
   ↪ calldata _blsPublicKey) external onlyModule {
```

```
28     address _stakeHouse,
```

```
46 function _onlyValidStakeHouse(address _stakeHouse) internal view
   ↪ override {
```

```
52 function _onlyStakeHouseKnotThatHasNotRageQuit(address _stakeHouse,
   ↪ bytes calldata _blsPubKey) internal view override {
```

```
57 function _onlyValidStakeHouseKnot(address _stakeHouse, bytes
   ↪ calldata _blsPubKey) internal view override {
```

CVF-181. INFO

- **Category** Unclear behavior
- **Source** savETHRegistryDestinationGateway.sol

Description These functions should emit some events.

```
17 function setIsValidHouse(address _stakeHouse) external onlyModule {
```

```
22 function setAssociatedHouseForKnot(address _stakeHouse, bytes
   ↪ calldata _blsPublicKey) external onlyModule {
```

```
27 function rageQuitAndCleanup(
```

CVF-182. INFO

- **Category** Procedural
- **Source** savETHRegistryDestinationGateway.sol

Description These two functions are identical.

Recommendation Consider merging them into one, or implementing one through the other.

52
```
function _onlyStakeHouseKnotThatHasNotRageQuit(address _stakeHouse,
   ↪ bytes calldata _blsPubKey) internal view override {
```

57
```
function _onlyValidStakeHouseKnot(address _stakeHouse, bytes
   ↪ calldata _blsPubKey) internal view override {
```

CVF-183. INFO

- **Category** Suboptimal
- **Source** savETHGateway.sol

Recommendation This could be simplified as: import { StakeHouseUniverse } from ”./StakeHouseUniverseAPI.sol”;

6
```
import { StakeHouseUniverse } from "../../gateway-implementations/
   ↪ dETH/StakeHouseUniverseAPI.sol";
```

CVF-184. INFO

- **Category** Procedural
- **Source** savETHGateway.sol

Description Declaring top-level errors in a file named after a contract makes it harder to navigate through code.

Recommendation Consider either moving the errors into the contract or moving them into a separate file.

15
```
error OnlyIndexOwner();
error InvalidBLSPubKey();
error KNOTIsNotInAnIndex();
error AlreadyMigrated();
error InvalidIndexId();
```

CVF-185. INFO

- **Category** Bad datatype
- **Source** savETHGateway.sol

Recommendation The type of this argument should be ”StakeHouseUniverse”.

```
32 address _universe,
```

CVF-186. INFO

- **Category** Bad datatype
- **Source** savETHGateway.sol

Recommendation The type of this argument should be ”ConsentVerification”.

```
35 address _consentVerificationModule,
```

CVF-187. INFO

- **Category** Bad datatype
- **Source** savETHGateway.sol

Recommendation The type of this argument should be ”RPBSVerificationLibrary”.

```
36 address _rpbsVerificationModule,
```

CVF-188. INFO

- **Category** Bad datatype
- **Source** savETHGateway.sol

Recommendation The type of these arguments could be more specific.

```
47 address _stakeHouse,
```

```
51 address _originGateway,
```

```
53 address _destinationGateway
```

CVF-189. INFO

- **Category** Bad datatype
- **Source** savETHGateway.sol

Recommendation The type of the ”stakeHouse” returned value should be more specific.

```
75 ) internal override view returns (address stakeHouse, uint256
     ↪ gatewayIndex, bytes memory blsPubKey) {
```

CVF-190. INFO

- **Category** Bad datatype
- **Source** savETHGateway.sol

Recommendation The value ”48” here should e a named constant.

```
91 if (_baseParams.paramThree.length != 48) revert InvalidBLSPubKey();
```

CVF-191. INFO

- **Category** Bad datatype
- **Source** savETHGateway.sol

Recommendation The value ”24 ether” should be a named constant.

```
99 if (knotDETHBalanceInIndex < 24 ether) revert InvalidKnotBalance();
```

CVF-192. INFO

- **Category** Suboptimal
- **Source** savETHGateway.sol

Recommendation If no precision loos is guaranteed here, consider adding an explicit assert.

```
104 // As per original deposit contract and beacon chain balances, we
      ↪ process amounts in gwei with no procesion loss.
```

CVF-193. INFO

- **Category** Suboptimal
- **Source** StakeHouseUniverseAPI.sol

Recommendation This contract could be turned into an interface.

```
9 abstract contract StakeHouseUniverse {
```

CVF-194. INFO

- **Category** Bad datatype
- **Source** StakeHouseUniverseAPI.sol

Recommendation The returned type should be more specific.

```
21 function memberKnotToStakeHouse(bytes calldata _blsPubKey) external
      ↪ view virtual returns (address);
```

CVF-195. INFO

- **Category** Bad datatype
- **Source** StakeHouseUniverseAPI.sol

Recommendation The type of these arguments should be more specific.

```
25 address stakeHouse,     // Address of registered StakeHouse
   address sETHAddress,    // Address of sETH address associated with
      ↪ StakeHouse
```

CVF-196. INFO

- **Category** Procedural
- **Source** RPBSVerificationLibrary.sol

Description Despite the name this is a contract rather than a library.

Recommendation Consider renaming or turning into a library.

```
12 contract RPBSVerificationLibrary is IRBSEndorserSignature,
      ↪ IAccumulatorMetadata, IGatewayStructs, Initializable {
```

CVF-197. INFO

- **Category** Bad naming
- **Source** RPBSVerificationLibrary.sol

Description The semantics of the keys is unclear.

Recommendation Consider elaborating more on this.

```
21 mapping(uint256 => mapping(string => bool)) public
      ↪ isUTXOEndorsedByPublicKey;
```

CVF-198. INFO

- **Category** Procedural
- **Source** RPBSVerificationLibrary.sol

Recommendation It is a good practice to put a comment into an empty block to explain why the block is empty.

```
24 constructor() initializer {}
```

CVF-199. INFO

- **Category** Suboptimal
- **Source** RPBSVerificationLibrary.sol

Recommendation The type of the ”_registry” argument should be ”EndorserRegistry” or an interface extracted from it.

```
26 function init(address _registry, uint256
      ↪ _numberOfEndorsementsRequired) external initializer {
```

CVF-200. INFO

- **Category** Suboptimal
- **Source** RPBSVerificationLibrary.sol

Description The function always returns true.

Recommendation Consider returning nothing.

```
39 ) external virtual returns (bool) {
```

CVF-201. INFO

- **Category** Procedural
- **Source** RPBSVerificationLibrary.sol

Description The expression ”encodePointHex(_endorsements[i]” is calculated twice.

Recommendation Consider calculating once and reusing.

```
76 require(!isUTXOEndorsedByPublicKey[_depositMetadata.depositLeafIndex
   ↪ ][encodePointHex(_endorsements[i].publicKey)], "Already
   ↪ endorsed");
```

```
87 isUTXOEndorsedByPublicKey[_depositMetadata.depositLeafIndex][
   ↪ encodePointHex(_endorsements[i].publicKey)] = true;
```

CVF-202. INFO

- **Category** Documentation
- **Source** ConsentVerification.sol

Description The semantics of the parameters is unclear.

Recommendation Consider documenting.

```
14 "Consent(uint256,address,uint256,bytes)"
```

CVF-203. INFO

- **Category** Procedural
- **Source** ConsentVerification.sol

Recommendation It is a good practice to put a comment into am empty block to explain why the block is empty.

```
18 constructor() initializer {}
```

CVF-204. INFO

- **Category** Bad naming
- **Source** ConsentVerification.sol

Description The word ”validate” is commonly used for function that revert in case of invalid argument, while the word ”check” is used for function that return validity flag.

Recommendation Consider renaming to ”checkConsentSignature”.

```
50 function validateConsentSignature(
```

CVF-205. INFO

- **Category** Bad datatype
- **Source** AccumulatorFactory.sol

Recommendation The parameter type should be ”IAccumulator”.

```
10 event AccumulatorCloned(address indexed newAccumulator);
```

CVF-206. INFO

- **Category** Bad datatype
- **Source** AccumulatorFactory.sol

Recommendation The type of this variable should be ”Accumulator” or ”IAccumulator”.

```
13 address public implementation;
```

CVF-207. INFO

- **Category** Bad datatype
- **Source** AccumulatorFactory.sol

Recommendation The return type should be ”IAccumulator”.

```
20 function deployAccumulator(address _manager) external returns (
   ↪ address) {
```

CVF-208. INFO

- **Category** Suboptimal
- **Source** CommonErrors.sol

Recommendation These errors could be made more useful by adding some parameters to them.

```
error OnlyGateway();
error InvalidSummary();
error InvalidAmount();
error InvalidRecipient();
error LeafNotValidOnOrigin();
error InvalidOwner();
error InconsistentArrayLengths();
error ChainChanged();
error InvalidDestinationChainId();
error InvalidDestinationGatewayAddress();
error DepositTooHigh();
error InvalidOrigin();
error InvalidDomain();
error InvalidKnotBalance();
```

CVF-209. INFO

- **Category** Bad naming
- **Source** IAccountManager.sol

Description The semantics of the returned values is unclear.

Recommendation Consider giving descriptive names to the returned values and/or describing them in a documentation comment.

```
bytes memory,
bytes memory,
bool ,
uint64,
uint64,
uint64,
uint64,
uint64,
uint64
```

CVF-210. INFO

- **Category** Unclear behavior
- **Source** IAccountManager.sol

Description These functions should emit some events and these events should be declared in this interface.

```
50 function createStakehouse(
```

```
63 function joinStakehouse(
```

```
77 function joinStakeHouseAndCreateBrand(
```

```
89 function registerValidatorInitials(
```

```
99 function registerDeposit(
```

CVF-211. INFO

- **Category** Suboptimal
- **Source** IAccountManager.sol

Recommendation Short strings could be efficiently packed into ”byte32” values. https://gist.github.com/3sGgpQ8H/567354534170905e047b299286697e19

```
53 string calldata _ticker,
```

```
80 string calldata _ticker,
```

CVF-212. INFO

- **Category** Bad datatype
- **Source** IAccountManager.sol

Recommendation The return type should be more specific.

```
55 ) external returns (address stakehouse);
```

CVF-213. INFO

- **Category** Bad datatype
- **Source** IAccountManager.sol

Recommendation The type of these arguments should be more specific.

```
66 address _stakehouse,
```

```
81 address _stakehouse,
```

CVF-214. INFO

- **Category** Suboptimal
- **Source** IAccountManager.sol

Recommendation These parameters should be indexed.

Client Comment *Bytes cannot be indexed.*

```
109 bytes blsPubKey
```

```
114 bytes blsPubKey,
```

```
122 bytes blsPubKey
```

```
127 bytes blsPubKey
```

```
132 bytes blsPubKey
```

```
137 bytes blsPubKey
```

CVF-215. INFO

- **Category** Suboptimal
- **Source** ISavETHRegistry.sol

Recommendation The ”memberId” parameters should be indexed.

```
7  event KnotTransferredToAnotherIndex(bytes memberId, uint256 indexed
       ↪ newIndexId);

10 event ApprovedSpenderForKnotInIndex(bytes memberId, address indexed
       ↪ spender);

13 event KnotAddedToOpenIndex(bytes memberId, address indexed
       ↪ indexOwner, uint256 savETHSent);

16 event KnotAddedToOpenIndexAndDETHWithdrawn(bytes memberId);

19 event dETHAddedToKnotInIndex(bytes memberId, uint256 dETH);

25 event dETHReservesAddedToOpenIndex(bytes memberId, uint256 amount);

28 event RageQuitKnot(bytes memberId);

34 event KnotInsertedIntoIndex(bytes memberId, uint256 indexed indexId)
       ↪ ;
```

CVF-216. INFO

- **Category** Suboptimal
- **Source** ISavETHRegistry.sol

Recommendation The ”indexId” parameter should be indexed.

```
40 event IndexOwnershipTransferred(uint256 indexId);
```

CVF-217. INFO

- **Category** Bad datatype
- **Source** ISavETHRegistry.sol

Recommendation The type of these arguments should be more specific.

```
71  address _stakeHouse,
```

```
83  address _stakeHouse,
```

```
95  address _stakeHouse,
```

```
107 address _stakeHouse,
```

```
119 address _stakeHouse,
```

```
131 address _stakeHouse,
```

CVF-218. INFO

- **Category** Bad naming
- **Source** ISavETHManager.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or describing in a documentation comment.

```
8   function createIndex(address _owner) external returns (uint256);
```

CVF-219. INFO

- **Category** Unclear behavior
- **Source** ISavETHManager.sol

Description These functions should emit some events and these events should be declared in this interface.

```
8  function createIndex(address _owner) external returns (uint256);
13 function approveForIndexOwnershipTransfer(
21 function transferIndexOwnership(uint256 _indexId, address _to)
       ↪ external;
27 function transferKnotToAnotherIndex(
37 function approveSpendingOfKnotInIndex(
47 function addKnotToOpenIndex(
57 function isolateKnotFromOpenIndex(
67 function addKnotToOpenIndexAndWithdraw(
77 function depositAndIsolateKnotIntoIndex(
85 function withdraw(address _recipient, uint128 _amount) external;
89 function deposit(address _recipient, uint128 _amount) external;
```

CVF-220. INFO

- **Category** Bad datatype
- **Source** ISavETHManager.sol

Recommendation The type of these arguments should be more specific.

```
28 address _stakeHouse,
```

```
38 address _stakeHouse,
```

```
48 address _stakeHouse,
```

```
58 address _stakeHouse,
```

```
68 address _stakeHouse,
```

```
78 address _stakeHouse,
```

CVF-221. INFO

- **Category** Bad datatype
- **Source** ISavETHManager.sol

Recommendation The return type should be more specific.

```
128 function dETHToken() external view returns (address);
```

```
131 function savETHToken() external view returns (address);
```

CVF-222. INFO

- **Category** Procedural
- **Source** IGatewayStructs.sol

Description This interface contains only structs.

Recommendation Consider turning it into a library or moving the structs to the top level and removing the interface.

```
7 interface IGatewayStructs is IEIP721Signature {
```

CVF-223. INFO

- **Category** Bad datatype
- **Source** IGatewayStructs.sol

Recommendation The type of these fields should be ”IGateway”.

```
20 address originGateway;
```

```
22 address destinationGateway;
```

CVF-224. INFO

- **Category** Bad naming
- **Source** IGateway.sol

Recommendation Events are usually named via nouns, such as ”Deployment”, ”Domain”, etc.

```
14 event Deployed();
```

```
17 event DomainAdded(uint256 indexed domainId);
```

```
20 event KillSwitchTriggered(uint256 indexed domainId);
```

```
23 event DomainCheckpointInjected(uint256 indexed domainId);
```

```
26 event DomainRecoveryMerkleRootInjected(uint256 indexed domainId);
```

```
29 event GatewayPaused();
```

```
32 event GatewayUnpaused();
```

```
35 event DomainPaused(uint256 indexed domain);
```

```
38 event DomainUnPaused(uint256 indexed domain);
```

```
41 event GatewayOperatorUpdated(address indexed operator, bool
   ↪ isEnabled);
```

```
44 event GatewayAdminUpdated(address indexed operator, bool isEnabled);
```

CVF-225. INFO

- **Category** Procedural
- **Source** IEIP721Signature.sol

Description EIP-712 describes format of messages to be signed, but doesn't describe format of signatures. Referring to EIP-712 here is misleading.

Recommendation Consider renaming to "IECDSASignature".

```
5 /// @notice Re-usable interface for sharing a common structure for
     ↪ defining an EIP712 signature
```

CVF-226. INFO

- **Category** Procedural
- **Source** IEIP721Signature.sol

Description This interface contains only structs.

Recommendation Consider turning it into a library or moving the structs to the top level and removing the interface.

```
6 interface IEIP721Signature {
```

CVF-227. INFO

- **Category** Bad naming
- **Source** IGatewayDispenser.sol

Description The semantics of the parameters is unclear.

Recommendation Consider giving descriptive names to the parameters and/or adding a documentation comment.

```
9 event Dispense(address, uint256, bytes);
```

CVF-228. INFO

- **Category** Unclear behavior
- **Source** IGatewayDispenser.sol

Description These functions should emit some events and these events should be declared in this interface.

```
21 function activateRecovery() external;
```

```
24 function injectForeignDomainCheckpoint(uint256
   ↪ _destinationRecoveryCheckpoint) external;
```

```
27 function injectRecoveryMerkleRoot(bytes32 _merkleRoot) external;
```

```
33 function dispenseViaRecovery(
```

CVF-229. INFO

- **Category** Bad naming
- **Source** IGatewayDispenser.sol

Description The semantics of the arguments and the returned values is unclear.

Recommendation Consider giving descriptive names to the arguments and the returned values and/or describing in documentation comments.

```
30 function dispense(uint256, address, uint256, bytes calldata)
   ↪ external returns (uint256);
```

```
42 function isPartOfMerkleTree(uint256, address, uint256, bytes
   ↪ calldata, bytes32[] calldata) external returns (bytes32, bool)
   ↪ ;
```

CVF-230. INFO

- **Category** Bad naming
- **Source** IGatewayDispenser.sol

Description The semantics of the returned value is unclear.

Recommendation Consider giving a descriptive name to the returned value and/or describing in a documentation comment.

```
39 ) external returns (uint256);
```

CVF-231. INFO

- **Category** Procedural
- **Source** IAccumulatorMetadata.sol

Description This interface only contains structs.

Recommendation Consider turning into a library or just moving the structs to the top level and removing this interface.

```
5 interface IAccumulatorMetadata {
```

CVF-232. INFO

- **Category** Bad naming
- **Source** IGatewayIngestor.sol

Recommendation Events are usually named via nouns, such as ”Consumption”.

```
8 event Consume(address, uint256, bytes);
```

CVF-233. INFO

- **Category** Suboptimal
- **Source** IAccumulator.sol

Description This function should return the leaf index.

```
10 function injectVectorCommitment(bytes32 _node) external;
```

CVF-234. INFO

- **Category** Unclear behavior
- **Source** IAccumulator.sol

Description This function should emit some event and this event should be declared in this interface.

```
10 function injectVectorCommitment(bytes32 _node) external;
```

CVF-235. INFO

- **Category** Bad naming
- **Source** IAccumulator.sol

Description The interface is able to deal with arbitrary commitments, but here the word ”deposit” refers to a particular use case.

Recommendation Consider renaming to ”count”.

```
13 function depositCount() external view returns (uint256);
```

CVF-236. INFO

- **Category** Suboptimal
- **Source** IAccumulator.sol

Description This function is superseded by the ”getAccumulatorRootAndCount” function.

Recommendation Consider refactoring.

```
13 function depositCount() external view returns (uint256);
```

ABDK
Consulting

About us

Established in 2016, is a leading service provider in the space of blockchain development and audit. It has contributed to numerous blockchain projects, and co-authored some widely known blockchain primitives like Poseidon hash function.

The ABDK Audit Team, led by Mikhail Vladimirov and Dmitry Khovratovich, has conducted over 40 audits of blockchain projects in Solidity, Rust, Circom, C++, JavaScript, and other languages.

Contact

Email

dmitry@abdkconsulting.com

Website

abdk.consulting

Twitter

twitter.com/ABDKconsulting

LinkedIn

linkedin.com/company/abdk-consulting