Quantstamp

Sequence - Identity Document

Executive Summary

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

Type	Off-Chain signer
Timeline	2025-07-08 through 2025-07-28
Language	Solidity
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review, Penetration Testing
Specification	Internal Documentation
Source Code	• 0xsequence/identity-instrument #7bd5446
Auditors	• Nikita Belenkov Senior Auditing Engineer • Leonardo Passos Senior Research Engineer • Albert Heinle Auditing Engineer

Documentation quality	High
Test quality	Medium
Total Findings	46 Fixed: 25 Acknowledged: 21
High severity findings	9 Fixed: 9
Medium severity findings	15 Fixed: 14 Acknowledged: 1
Low severity findings	6 Fixed: 2 Acknowledged: 4
Undetermined severity findings	0
Informational findings	16 Acknowledged: 16

Summary of Findings

Quantstamp has performed an audit and a penetration test of Sequence's Identity Instrument. The Sequence Identity Instrument is a Go-based authentication and identity management system that provides enclave-based authentication using multiple methods, including OIDC, OAuth 2.0, and email OTP. The system uses AWS Nitro Enclaves for cryptographic operations and implements Shamir's Secret Sharing for key management across trusted third parties (TTPs).

During the audit, a range of issues have been identified, mainly revolving around the penetration testing side of the engagement. However, a few issues in the design and authentication implementation have also been identified. We recommend that all the issues identified in the report be addressed.

The test suite is quite limited and should be improved.

Fix review 1 (November 12th, 2025)

Many issues remain unresolved and must be promptly addressed. Non-fixed issues center on insecure JWT handling, missing schema/input restrictions, and weak validation on several endpoints. Some replay-attack risks and incomplete error handling remain. A few infrastructure and key-rotation items are only partially mitigated.

Latest commit hash: 129bdf2975ddb7834a212ecd48c2df93a259cb33

Fix review 2 (December 1st, 2025)

Most of the outstanding issues of the first fix-review phase have been addressed. A small subset was classified as acknowledged, indicating that while the behavior exists, it does not present a material security or functional risk in the current context. No issues remain unresolved. Tests have also improved in comparison to the initial audit. The mean statement coverage is presently at 71%.

The project’s test coverage shows strong emphasis on security-critical components—such as encryption, OTP flows, OIDC validation, and telemetry—which consistently achieve high coverage and indicate solid engineering practices. However, important modules, including AuthCode and ID-Token handlers, metadata validation, and parts of the RPC layer, exhibit only partial coverage, leaving error paths and boundary conditions insufficiently tested. Some operational packages, particularly CLI entrypoints and configuration loading, have no coverage at all, increasing the risk of deployment-time issues. Overall, while the core authentication and cryptographic layers are well tested, the project would benefit from broader coverage of negative cases, RPC edge behavior, and initialization logic.

Latest commit hash: 27f658cb30c63a3c2c265c570049bdd83e246045

ID	DESCRIPTION	SEVERITY	STATUS
SEQ-ID-1	JWT Token Validation and Verification Are Explicitly Disabled	● High	Fixed
SEQ-ID-2	High Severity CVE Present in Code Base	● High	Fixed
SEQ-ID-3	Issuers in JWT tokens should never allow for unencrypted traffic	● High	Fixed
SEQ-ID-4	Recommending to Sanitize Input and Use Multi-Tier Architecture Instead of Passing Down the HTTP Context to Every Function	● High	Fixed
SEQ-ID-5	Enable Deletion Protection for Load Balancers	● High	Fixed
SEQ-ID-6	Set Immutable Tags for Docker Images Stored in Ecr	● High	Fixed
SEQ-ID-7	Enable Image Scanning for Docker Images Stored in Ecr	● High	Fixed
SEQ-ID-8	Avoid Legacy EC2 Metadata Format	● High	Fixed
SEQ-ID-9	Missing Key Generation Update Logic	● High	Fixed
SEQ-ID-10	A User Is Able to Pass in an Empty Nonce Value	● Medium	Fixed
SEQ-ID-11	No CI/CD Tool in Place for Pipeline Tests	● Medium	Fixed
SEQ-ID-12	No locked in version in Dockerfile	● Medium	Fixed
SEQ-ID-13	Use Encryption at Rest with Dynamodb Tables	● Medium	Fixed
SEQ-ID-14	Drop Invalid Headers on the Load Balancer Level	● Medium	Fixed
SEQ-ID-15	Enable Enhanced Monitoring on EC2 Instances	● Medium	Fixed
SEQ-ID-16	Ensure to Have Termination Protection for EC2 Instances	● Medium	Fixed
SEQ-ID-17	Auto-Redirect to Auth Does Not Work for Invalid Subdomains	● Medium	Fixed
SEQ-ID-18	Unhandled Errors in Code	● Medium	Fixed
SEQ-ID-19	DynamoDB Table Safety Considerations with Respect to Current Design	● Medium	Fixed
SEQ-ID-20	Low Number of Available TTPs Can Compromise System Availability	● Medium	Acknowledged
SEQ-ID-21	Auth Session Could be Abused and Lead to DoS Attacks	● Medium	Fixed
SEQ-ID-22	Lack of Email and Length Validation Poses Security Risks	● Medium	Fixed
SEQ-ID-23	No Way to Rotate Underlying Private Keys	● Medium	Fixed
SEQ-ID-24	Auth Commitment Overwrite Allows DoS on User Authentication	● Medium	Fixed
SEQ-ID-25	Object String Representations with Character Delimiters Need to Be Reconsidered	● Low	Acknowledged

ID	DESCRIPTION	SEVERITY	STATUS
SEQ-ID-26	Set Retention Period for Cloudwatch Logs	● Low	Acknowledged
SEQ-ID-27	Use Builder Libraries for Special Strings	● Low	Fixed
SEQ-ID-28	Signed Messages Are Prone to Replay Attacks	● Low	Fixed
SEQ-ID-29	Email-based OTP Lack of Limits May Lead to Spam and Potential DoS Attacks	● Low	Acknowledged
SEQ-ID-30	OTP-Based Email Subject May Unnecessarily Expose Users	● Low	Acknowledged
SEQ-ID-31	Docker Containers Run Their Commands as Root	● Informational	Acknowledged
SEQ-ID-32	Serve OWASP Recommended Headers with Each Request	● Informational	Acknowledged
SEQ-ID-33	Capture and Persist Operating System Logs Past the Lifetime of a Container	● Informational	Acknowledged
SEQ-ID-34	Docker Compose Requires Logging Size Protection	● Informational	Acknowledged
SEQ-ID-35	Pin Versions of Packages Installed in a Docker Image	● Informational	Acknowledged
SEQ-ID-36	Set Limited Capabilities for Containers in Docker Compose	● Informational	Acknowledged
SEQ-ID-37	Do Not Mount the Host's `docker.sock` for Any Docker Container	● Informational	Acknowledged
SEQ-ID-38	Enable Bucket Logging for S3 Buckets	● Informational	Acknowledged
SEQ-ID-39	Bind HTTP Server to a Specific Network Interface	● Informational	Acknowledged
SEQ-ID-40	Docker-Compose Should Have a Restart Policy Specified	● Informational	Acknowledged
SEQ-ID-41	Have Healthcheck Parameters in Dockerfiles	● Informational	Acknowledged
SEQ-ID-42	Ensure to Have a Backup Plan for Ec2 Instances	● Informational	Acknowledged
SEQ-ID-43	Enable Versioning on S3 Buckets	● Informational	Acknowledged
SEQ-ID-44	Root Port Usage Increases Attack Surface	● Informational	Acknowledged
SEQ-ID-45	Consider if Auth Key Maximum Ttl (90 Days) Is Needed	● Informational	Acknowledged
SEQ-ID-46	Non-Secp256k1 Signers Should Be Rejected Early On	● Informational	Acknowledged

Assessment Breakdown

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

Disclaimer
Only features that are contained within the repositories at the commit hashes specified on the front page of the report are within the scope of the audit and fix review. All features added in future revisions of the code are excluded from consideration in this report.

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
- Infrastructure Misconfigurations.
- Go Project specific vulnerabilities.
- OWASP Top 10 (general and client-side).
- Full-lifecycle security, from build to deployment.
- SBOM and CVE vulnerabilities.
- Authentication/Authorization Errors
- Server-Side request forgery
- Client-Side request forgery
- DDos Prevention
- Injection Prevention

Methodology

1. Code review that includes the following
   1. Review of the specifications, sources, and instructions provided to Quantstamp to make sure we understand the size, scope, and functionality of the smart contract.
   2. Manual review of code, which is the process of reading source code line-by-line in an attempt to identify potential vulnerabilities.
   3. 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:
   1. 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.
   2. 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, clarity, 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.
5. Review of all configuration files in your infrastructure, coming from both code repositories and direct API connections to the cloud provider.
6. Apply CIS, DoD STIG and other applicable benchmarks to discovered configurations.
7. Review Architectural choices and determine if all required infrastructure components are present.

Scope

Only `auth/ cmd/ config/ data/ encryption/ etc/ o11y/ proto/ rpc/` folders were in scope, with everything else out of scope and assumed to be functioning correctly, including but not limited to:

- Vendor packages used, including `0xSequence` dependencies.

Operational Considerations

- As the name indicates, Trusted Third Party (TTPs) are trustworthy.
- Following industry best practices, it is assumed that access to the dynamodb instance is secure, with proper access controls. Furthermore, all stored items are encrypted.
- There is an assumption that KMS remote keys are properly managed by TTPs, as per industry best practices. It is also assumed that these TTPs follow strict security guidelines, which are checked and enforced from time-to-time to ensure overall system security.
- It is assumed that TTPs set up their KMS keys with policies that only allow access to the key if the correct EIF of the identity instrument is deployed, as identified by its measurements.
- It is assumed that all dependencies from the identity instrument, including those from 0xsequence itself, are secure.
- Shares are assumed to be communicated over an encrypted channel, as otherwise the underlying key can be easily recovered.
- Careful attention needs to be paid to the selection of OIDC Providers.

Findings

SEQ-ID-1 JWT Token Validation and Verification Are Explicitly Disabled

● High ⓘ Fixed

Update
Fixed in f26e15d669062d5da5a65f517e71346ab88d2a21

File(s) affected: `auth/idtoken/auth_handler.go`

Description: `identity-instrument-master/auth/idtoken/auth_handler.go`, line 108: The application disables both signature verification and claims validation when parsing a JWT token:

```
jwt.Parse([]byte(idToken), jwt.WithVerify(false), jwt.WithValidate(false))
```

This effectively turns off all critical security checks provided by JWTs, allowing any token—malicious or malformed—to be accepted as valid. An attacker can forge tokens with arbitrary claims, including user identifiers, roles, and permissions, potentially resulting in full account takeover, privilege escalation, or bypass of authentication.

JWT's purpose is to assert identity and claims integrity via cryptographic signatures. Disabling .WithVerify(false) skips this integrity check, while .WithValidate(false) disables checks like expiration (exp) or issuer (iss). Combined, this configuration nullifies the entire trust model of JWTs.

Recommendation: Re-enable both verification and validation when parsing JWTs:

```
jwt.Parse([]byte(idToken), jwt.WithVerify(true), jwt.WithValidate(true))
```

Ensure that:

- Tokens are signed with a secure algorithm (e.g., RS256, ES256 — avoid none, HS256 unless you fully control the secret).
- Issuer (iss), audience (aud), and expiration (exp) claims are validated against expected values.
- Any external JWTs (e.g., from Auth0, Cognito, or other IdPs) are validated against their respective public keys using JWKS.

Additionally:

- Write test cases that validate signature tampering and claim expiration.
- Review audit logs to detect if this issue may have been exploited prior to remediation.

SEQ-ID-2 High Severity CVE Present in Code Base

● High ⓘ Fixed

Update
Fixed in 1101da1450dba23d407284ba33d2ee287ebbc95d

Description: The system includes a dependency that is affected by CVE-2025-22869, a high-severity vulnerability. This CVE refers to a known flaw in the `golang.org/x/crypto`.

Use of vulnerable third-party components introduces serious risks, as they may be exploited even without a direct flaw in the application's own code. Attackers often scan systems for known vulnerable versions, especially those that are widely used.

Failure to patch or update this dependency may violate basic supply chain security principles and potentially leave the system open to compromise.

Recommendation: Upgrade the affected dependency to a patched version that addresses CVE-2025-22869.

Additionally, document all third-party packages and their associated risk levels as part of a software bill of materials (SBOM), and use automated tools in your CI/CD pipeline to detect them before any build.

SEQ-ID-3 Issuers in JWT tokens should never allow for unencrypted traffic

● High ⓘ Fixed

Update
Fixed in 118d097fd0b232fdae88c659bdf0f718a8bdb9dd

File(s) affected: `auth/idtoken/auth_handler.go`

Description: The OpenID Connect issuer (iss) used in the authentication flow is configured with an insecure `http://` scheme, violating the OIDC specification and creating a critical security risk.

According to the OpenID Connect Core 1.0 specification, the issuer MUST use the https scheme to ensure integrity, authenticity, and confidentiality of identity tokens and discovery documents. Using plain http opens the door to MITM (man-in-the-middle) attacks, token forgery, and discovery document tampering, especially if DNS spoofing or network interception is possible.

Even if the server is internal or behind a VPN, relying on http for OIDC interactions undermines the security model of federated identity, which assumes cryptographically verifiable trust boundaries.

Recommendation:

- Update the OIDC provider and client configurations to ensure the issuer URL begins with https://.
- Enforce TLS at the identity provider endpoint and ensure all metadata and token endpoints are also secured over HTTPS.
- Reject http:// issuer URLs in configuration or validation code to prevent future regressions.
- Re-test the authentication flow after the change to ensure no certificate validation or redirect issues remain.

SEQ-ID-4 Recommending to Sanitize Input and Use Multi-Tier Architecture Instead of Passing Down the HTTP Context to Every Function

● High ⓘ Fixed

Update

Fixed in 071b4228f004b1922cfc0d367adb730422313f1e

Description: The HTTP context of API calls contains all external, user-provided information. By nature, this data cannot be trusted and needs to be validated against tampering. The most common design for APIs consists of a resource layer, which checks the inputs for validity and passes down the sanitized inputs for further processing. The context itself should not be passed further down.

Recommendation: Ensure that there is a middleware that takes the inputs of an API and passes them further down to other operations in a sanitized manner.

The common name for such an architecture is the Multi-tier architecture.

SEQ-ID-5 Enable Deletion Protection for Load Balancers

● High ⓘ Fixed

Update

Fixed in c700a028be8881d361101dec3d462954cf2c4f9e

Description: Load balancers should be protected from accidental deletion.

Recommendation: For every resource of type `aws_lb`, ensure that the `enable_deletion_protection` key is set to `true` (default `false`).

Sources:

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/lb

SEQ-ID-6 Set Immutable Tags for Docker Images Stored in Ecr

● High ⓘ Fixed

Update

Fixed in c700a028be8881d361101dec3d462954cf2c4f9e

File(s) affected: `./ca-central-1/aws/ecr/ecr\_repository.tf`

Description: In order to maintain stability with respect to image tags and versions, it is important to set the tags in ECR to be immutable.

Recommendation: For every resource of type `aws_ecr_repository`, ensure that the `image_tag_mutability` is set to `IMMUTABLE` (default is `MUTABLE`).

Sources:

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/ecr_repository

SEQ-ID-7 Enable Image Scanning for Docker Images Stored in Ecr

● High ⓘ Fixed

Update

Fixed in c700a028be8881d361101dec3d462954cf2c4f9e

Description: The ECR container registry should do a vulnerability scan on every pushed image.

Recommendation: For every resource of type `aws_ecr_repository`, ensure that the `image_scanning_configuration` block exists and has its key `scan_on_push` set to `true`.

Sources:

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/ecr_repository

SEQ-ID-8 Avoid Legacy EC2 Metadata Format

● High ⓘ Fixed

Update

Fixed in cde6c2b4e192ab59714c8e53dbf823f9f4db6698

Description: The EC2 instances are configured to use Instance Metadata Service version 1 (IMDSv1), which is susceptible to server-side request forgery (SSRF) and other local privilege escalation attacks. IMDSv1 is a request/response interface accessible via HTTP and lacks the session-based protections introduced in IMDSv2. Because it allows for unauthenticated HTTP requests from any process running on the instance, an attacker with the ability to execute code or exploit SSRF vulnerabilities can retrieve sensitive metadata — including IAM role credentials — from the metadata service.

Recommendation: Enforce the use of IMDSv2 by updating the instance metadata options:

- Set `HttpTokens=required`
- Set `HttpEndpoint=enabled`

SEQ-ID-9 Missing Key Generation Update Logic

● High ⓘ Fixed

Update

Fixed in 73c69155257522703193b576e621eb1768f76954

File(s) affected: `encryption/pool.go`

Description: The Decrypt function in `encryption/pool.go` contains a TODO comment indicating that key generation should be updated if needed, but this logic is not implemented. When a key from an older generation is used for decryption, the system should automatically update it to the current generation to ensure it remains accessible as TTPs rotate their keys. The system does not check if the key's generation is current, nor does it attempt to update the key to the latest generation. If keys cannot be updated to the current generation, users may lose access to their encrypted data.

Recommendation:

- Implement the key generation update logic after successful decryption.
- Check if the key's generation is current, and if not, update it to the latest generation.
- Add monitoring and alerting for key generation updates to track the health of the key management system.

SEQ-ID-10 A User Is Able to Pass in an Empty Nonce Value

● Medium ⓘ Fixed

Update

Fixed in fcc3bbce2115cfdcb85612c36aec565ba04d046f

File(s) affected: `rpc/internal/attestation/middleware.go`

Description: The application contains logic that checks if the value of a variable nonceVal is equal to an empty string (`identity-instrument-master/rpc/internal/attestation/middleware.go`, line 69) before performing further operations. However, if the input (Header `"X-Attestation-Nonce"`) is a string containing only whitespace (e.g., " "), the check incorrectly allows the code to continue, as it only excludes exactly empty strings.

This leads to a scenario where malformed or unintended input values—such as whitespace—are treated as valid. In security-sensitive contexts, such as nonce verification, this can result in logic flaws, inconsistent behavior, or bypass conditions.

Recommendation: Update the check to account for trimmed whitespace.

Incorporate unit tests that cover edge cases like whitespace-only input.

Where applicable, consider stricter validation rules (e.g., regex validation, length checks, encoding enforcement).

SEQ-ID-11 No CI/CD Tool in Place for Pipeline Tests

● Medium ⓘ Fixed

Update

Fixed in 2870502b9e307f807edc5ac6994f927c57197d8a (added test & build CI actions) and 777d7df07c706224e0b20a8a83af86e110e96e5f (added Lint CI action)

Description: In order to prevent issues when deploying changes to the code, it is recommended to employ a so-called CI/CD tool. Examples are:

- Github Workflows
- Jenkins
- Bitbucket pipelines
- Gitlab pipelines
- CircleCI

These tools can be used to ensure that a proper code change process is in place and that tests are run every time a change takes place.

Recommendation: Employ a CI/CD tool, and at the very least, instruct it to run checks on pull requests to your repository.

Sources:
- https://www.nist.gov/cyberframework

SEQ-ID-12 No locked in version in Dockerfile

● Medium ⓘ Fixed

Update
Client's response:

```
The Dockerfile FROM directives are already locked to specific digests forboth the Go builder and the
Eiffel EIF build tool.
```

File(s) affected: `Dockerfile`

Description: When using `FROM` directives in Docker files, it is important to specify the version directly and not just use latest (since this is a changing tag). Alternatively, define a specific digest of the image you aim to use.

Recommendation: For any `FROM` command, ensure that the image has a version or a digest specified and that the version does not contain `latest`.

Sources:
- https://docs.docker.com/engine/reference/builder/

SEQ-ID-13 Use Encryption at Rest with Dynamodb Tables

● Medium ⓘ Fixed

Update
Fixed in c700a028be8881d361101dec3d462954cf2c4f9e

Description: When defining a DynamoDB table, one needs to ensure that the data on the server is encrypted.

Recommendation: For every resource of type `aws_dynamodb_table`, ensure that the `enabled` key in the `server_side_encryption` block is set to `true` (default is `false`).

Sources:
- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/dynamodb_table

SEQ-ID-14 Drop Invalid Headers on the Load Balancer Level

● Medium ⓘ Fixed

Update
Fixed in c700a028be8881d361101dec3d462954cf2c4f9e

Description: A variety of attacks are actually performed using invalid symbols in headers. There is a setting in the load balancer, if it is an application load balancer, to prevent those headers to be forwarded to the target resource.

Recommendation: For every resource of type `aws_lb`, if the `load_balancer_type` is `application` (default), then ensure that the setting `drop_invalid_header_fields` is set to `true` (default is `false`).

Sources:
- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/lb

SEQ-ID-15 Enable Enhanced Monitoring on EC2 Instances

● Medium ⓘ Fixed

Update
Fixed in c700a028be8881d361101dec3d462954cf2c4f9e

Description: When using EC2-instances, if real-time view on data is critical, it is advisable to enable enhanced monitoring. This also enables teams to get alarms more timely.

Recommendation: For every resource of type `aws_instance`, ensure that the `monitoring` parameter is set to `true` (default `false`).

Sources:
- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance
- https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/using-cloudwatch-new.html

SEQ-ID-16
Ensure to Have Termination Protection for EC2 Instances

● Medium ⓘ Fixed

Update
Fixed in c700a028be8881d361101dec3d462954cf2c4f9e

Description: When using EC2-instances, it is advisable to enable termination protection, since it is otherwise possible to accidentally lose data or put the cluster in an undesired state.

Recommendation: For every resource of type `aws_instance`, ensure that the `disable_api_termination` parameter is set to `true`.

Sources:
- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance

SEQ-ID-17
Auto-Redirect to Auth Does Not Work for Invalid Subdomains

● Medium ⓘ Fixed

Update
Client's response:

```
The webapp router structure has been changed, this issue is no longer present.
```

Important: No commit is provided here, as this part of the audit was primarily conducted through pentesting.

Description: The application renders parts of the authenticated user interface (UI) to unauthenticated users who visit arbitrary or deep-linked paths (e.g., https://v3.sequence-dev.app/foo). While backend API calls fail due to lack of authentication, the presence of elements like "Logout" and partial dashboard components implies that routing and view rendering are not properly gated by authentication status.

This exposes internal UI logic and workflows, which can aid attackers in reconnaissance, phishing, or crafting targeted social engineering attacks. In more severe scenarios, it may also leak static content or mislead users into believing they are logged in.

Even if no sensitive data is exposed directly, showing logged-in-only UI to unauthenticated users violates the principle of least privilege and secure defaults.

Recommendation:
- Enforce a global authentication check at the front-end routing level.
- Redirect unauthenticated users attempting to access protected routes to the login page.
- Backend APIs should remain protected (as they currently are) — but front-end routes must also clearly separate public and private content.
- Consider server-side rendering environments or static site protections (e.g., via NGINX, Apache, or CloudFront behaviors) to avoid preloading protected components in any unauthenticated state.

SEQ-ID-18 Unhandled Errors in Code

● Medium ⓘ Fixed

Update
Fixed in 777d7df07c706224e0b20a8a83af86e110e96e5f

File(s) affected: `auth/authcode/auth_handler.go`, `cmd/ingress-proxy/main.go`, `data/cipher_key.go`, `encryption/pool.go`, `proto/identity-instrument.gen.go`, `rpc/internal/attestation/middleware.go`, `rpc/internal/attestation/userdata.go`, `rpc/auth.go`

Description: Several places in the application lack proper error handling, where errors are either ignored silently, logged without context, or not acted upon. This practice leads to undefined behavior, unstable execution paths, and security blind spots, especially if these failures impact authentication, access control, data validation, or external service communication.

Uncaught errors can:

- Suppress indicators of failed security checks
- Lead to silent partial failures or inconsistent states
- Open up denial-of-service conditions or privilege escalation pathways when failure isn't managed

Below is a list of instances where error handling was found to be missing or inadequate:

- `/identity-instrument-master/auth/authcode/auth_handler.go`, line 270: The function `jwtKey.Set` can also produce an error, which is not explicitly checked for in this function.
- `/identity-instrument-master/cmd/ingress-proxy/main.go`, line 30
- `/identity-instrument-master/data/cipher_key.go`, line 53: The `Write` function is not explicitly checked for an error that may be produced.
- `/identity-instrument-master/encryption/pool.go`, line 219.
- `/identity-instrument-master/proto/identity-instrument.gen.go`, lines 554, 541, 540 and 528.
- `/identity-instrument-master/rpc/internal/attestation/middleware.go`, line 64.
- `/identity-instrument-master/rpc/internal/attestation/userdata.go`, line 17.
- `/identity-instrument-master/rpc/auth.go`, line 369.

Recommendation:

- Review all identified locations and ensure errors are checked, logged meaningfully, and handled appropriately.
- Adopt consistent practices for error handling throughout the codebase, such as:
  - Returning and propagating errors instead of ignoring them
  - Using structured logging with context for operational visibility
  - Failing safe (e.g., deny access, abort transaction) on unexpected behavior
- Integrate linters or static analysis tools into CI pipelines to catch these automatically.

SEQ-ID-19
DynamoDB Table Safety Considerations with Respect to Current Design

● Medium ⓘ Fixed

Update

Fixed in 7f836826249f9642979088e287b34b6ab85fd7fc

Client's response:

```
Some of the data (signers and cipher_keys tables) is permanent; these tables do not have TTL
configured. The other tables (auth_keys and auth_commitments) contain temporary data with TTL
configured. Regarding item size, all items have a known maximum size, which is why no advanced
partitioning solution is needed. The new tests ensure that this is the case. In the unforeseen scenario
that an item is larger than DynamoDB limit, it is preferred that an error is returned, as it is either
a malicious action or misconfiguration.
```

Description: There are two critical considerations in the way the application interacts with Amazon DynamoDB, especially in the context of permanent data storage and unbounded payload sizes:

A. Time-to-Live (TTL) Not Explicitly Controlled

DynamoDB offers an optional Time to Live (TTL) mechanism that allows automatic deletion of expired items, based on a user-defined timestamp attribute.

If the application intends to store items permanently, it is vital that:

- No TTL attribute (e.g., ttl or expirationTime) is accidentally set.
- Data written into the table is validated to ensure TTL values are not unintentionally populated.

If TTL is enabled without strict control, this can lead to silent data deletion, resulting in:

- Unexpected missing records
- Failures in business logic
- Irrecoverable loss of important data (DynamoDB TTL deletions are permanent)

B. Risk of Exceeding the 400 KB Item Limit

DynamoDB enforces a hard 400 KB size limit per item (including attribute names and metadata). If the data model allows unbounded growth — such as appending to a list or storing user-generated content — there's a real risk of:

- Write failures (ValidationException: Item size has exceeded the maximum allowed size)
- Application crashes or inconsistencies due to write failures
- Inability to restore or replicate oversized items

This is especially critical when handling logs, document snapshots, or unvalidated client input.

Recommendation: For TTL Misconfiguration:

- Audit the schema to ensure that no TTL attribute is being unintentionally written.
- If permanent storage is desired:
  - Disable TTL on the table entirely via the AWS Console or CLI.
  - OR ensure your application never sets the TTL field unless explicitly intended.

To disable TTL:

Use a static type system or schema validation (e.g., JSON schema, Go structs with tags) to ensure TTL attributes are only set in specific contexts.

For 400 KB Item Limit: Implement automated tests or validation logic to reject writes of objects larger than ~350 KB (buffered to account for attribute names and metadata).

- Use structured data sharding:
  - Split large items into multiple records indexed by the same partition key with a range (e.g., part-001, part-002)
  - OR offload large payloads to Amazon S3, and store a reference in DynamoDB.
- Monitor application logs and CloudWatch metrics for ValidationException errors tied to item size.

SEQ-ID-20
Low Number of Available TTPs Can Compromise System Availability

● Medium ⓘ Acknowledged

Update

Client's response:

```
The initial decision to onboard 2 additional Trusted Third Parties was based on internal risk
assessment. More TTPs might be onboarded in the future based on changing risk assessment and outlook on
currently onboarded TTPs. This is because another TTP does not need to be onboarded immediately after
one of the currently active ones becomes unavailable. In such a scenario, if onboarding a new TTP
quickly is not possible, we can temporarily modify the threshold to 1 (requiring only a single TTP) and
re-encrypt the keys. This would, temporarily, reduce the security of the data slightly. However, with
proper access management, this is still the better optionm, inorder to avoid potential data loss in a
disaster scenario
```

File(s) affected: `encryption/pool.go`

Description: Currently, 0xsequence operates with three TTPs (itself, and two others), requiring two to be available at any time for encryption/decryption operations to occur successfully. However, this could be a risky situation, for example, if one TTP is out of the market unexpectedly, the time to onboard a new TTP could be lengthy. In the meantime, no other TTP could become unavailable.

Recommendation: Onboard at least four TTPs to have enough redundancy in case TTPs unexpectedly become unavailable.

SEQ-ID-21
Auth Session Could be Abused and Lead to DoS Attacks

● Medium ⓘ Fixed

Update

Fixed in cbef95436e4ca66562f49b9a3a04ba7bace67d19

File(s) affected: `rpc/sign.go`

Description: Upon a valid auth flow, one could abuse the sign logic by submitting many sign requests. Whenever a user session is in place, code-level rate limiting is highly advisable as an extra security layer to prevent Denial-of-Service (DoS) attacks, since cloud-based ones are less granular and generally based on IPs, which could be rotated.

Recommendation: Add code-level rate limiting based on any given active (non-expired) auth key. Stated otherwise, limit the number of sign requests that can be made per active auth key. Additionally:

- Consider using CAPTCHAs before one can trigger the whole OTP email auth flow.
- Log all attempts higher than the accepted rate limit to flag suspicious behavior.

SEQ-ID-22
Lack of Email and Length Validation Poses Security Risks

● Medium ⓘ Fixed

Update

Fixed in 46ca31b84b989f135cae021af15c018293cbaa8e

File(s) affected: `rpc/email/sender.go`, `rpc/sign.go`

Description: In the OTP authentication flow, in the commit stage, input email lengths are not validated. This creates an opportunity for malicious actors to send OTP requests with a large payload. This could open-up DoS attack vectors, such as Denial-of-Service (e.g., memory exaustion) or simply cause AWS costs to increase. Developers seem to be somewhat aware of this issue, as validation is currently stated as a TODO comment.

```
func (s \*Sender) NormalizeRecipient(recipient string) (string, error) {
    // TODO: Validate email address
    return Normalize(recipient), nil
 }
```

Similar to the missing email validation logic in the OTP-based auth flow, the sign logic does not verify the digest length, allowing for any digest size (generally expected to be 256 bytes long). The same issues concerning a potentially unbounded input apply here, namely, the risk of DoS attacks and potentially higher AWS costs.

Similar to the ones described above, the `auth_key` length should be validated as well.

Recommendation:
- Validate email inputs according to RFC 5321 limits.
- Validate digest and `auth_key` sizes, restricting them to the expected size.
- Sanitize and validate all user input server-side.
- Restrict the size of HTTP request payloads.
- Log input sizes (not values) that are unusually large to detect probing attacks.

SEQ-ID-23 No Way to Rotate Underlying Private Keys

● Medium Fixed

Update
Fixed in c3faa246b99daef3a471e10ea87a40b5e0a5312e

File(s) affected: `encryption/pool.go`

Description: The system does not provide a mechanism to rotate or replace the underlying private keys used for cryptographic operations. Key rotation is a critical security practice to limit the impact of key compromise and to comply with security best practices.

Recommendation:
- Implement a secure key rotation mechanism that allows for periodic or on-demand replacement of private keys.
- Ensure that old keys can be retired safely and that new keys are propagated to all necessary components.
- Document the rotation process and provide operational guidance for administrators.

SEQ-ID-24 Auth Commitment Overwrite Allows DoS on User Authentication

● Medium Fixed

Update
Fixed in c48c076c8f58033fb40c9e640777d58b0ff35eb9

File(s) affected: `rpc/auth.go`

Description: When initiating an auth flow, the identity instrument does not request any proof that one owns the auth keypair. The authkey parameter in any request simply represents the public key component of the keypair, and can be easily hijacked. An attacker could create a new auth flow with the a keypair that is already in use at the moment. This will allow the attacker to override the owner's auth key record, associating it with the attacker's signer.

The system, however, will not allow the attacker, nor the true owner, to proceed with signing transactions, causing a temporary DoS. The owner, in turn, will be forced to re-authenticate.

Recommendation: Similarly to what is currently done in the `Sign` logic, make the `Commit` stage require a proof that one actually owns the auth key. Make sure the signed message has a nonce to avoid replay attacks. From the signed message, recover the public key, using it to decrypt the message's content. Contrast the latter with the expected output, proceeding in the case of a match.

Another approach is not to allow the reuse of `auth_key` and reject any auth flows that are using an existing key.

SEQ-ID-25 Object String Representations with Character Delimiters Need to Be Reconsidered

• Low ⓘ Acknowledged

Update

Client's response

```
In most cases, these character-delimited strings are simple identifiers, created for comparison purposes, or to use as a key of a map, and never parsed. In others, the reason such strings are used is the need for a concise identifier for DynamoDB primary key (as SQL-like composite keys are not supported). For these reasons, a simple string is preferred over more complex structures and encodings.
```

Fix review notes:
We are marking this issue as acknowledged and decreasing its severity from "Medium" to "Informational".

File(s) affected: `data/signer.go`, `encryption/kms/remote_key.go`, `encryption/pool.go`, `proto/auth_id.go`, `proto/builder/builder.gen.go`, `proto/clients/identity-instrument.gen.go`, `proto/identity.go`, `proto/identity-instrument.gen.go`

Description: The codebase frequently represents complex objects or structured data as flat strings concatenated with single-character delimiters (e.g., `/`, `;`, or `:`). While this approach may seem lightweight, it introduces significant safety, parsing, and maintainability risks:

- There is no inherent escaping or encoding scheme, meaning data containing the delimiter (e.g., user input) can break parsing logic or introduce ambiguity.
- Any robust implementation would need to re-invent mechanisms for escaping, quoting, validation, and type coercion, which are notoriously error-prone and inconsistent.
- This approach increases the likelihood of data confusion bugs, silent logic failures, and even injection vulnerabilities in downstream systems if data is passed across system boundaries (e.g., logging, storage, network calls).

In some cases, the `strings.splitN` function is used, which then, in cases of the split-character appearing inside the strings itself, might cause a wrong un-marshalling.

Using delimiter-based string representations over structured formats is an anti-pattern, particularly when safe, standard, and well-supported alternatives like JSON are available. Modern libraries in all languages offer mature and efficient JSON serializers/deserializers, including type safety and schema validation.

The areas in the code where this occurred are:

- `data/signer.go`
- `encryption/kms/remote_key.go`
- `encryption/pool.go`
- `proto/auth_id.go`
- `proto/builder/builder.gen.go`
- `proto/clients/identity-instrument.gen.go`
- `proto/identity.go`
- `proto/identity-instrument.gen.go`

Recommendation: Replace ad hoc delimiter-based string representations with JSON-encoded representations of data structures.

Use official language-specific libraries (e.g., Go's encoding/json) to ensure safe, consistent parsing.

If delimiter-based formats must remain for compatibility reasons, enforce:

- `A rigorous escaping strategy`
- `Clear and centralized encoding/decoding logic`
- `Strong unit tests and fuzzing for corner cases`

SEQ-ID-26 Set Retention Period for Cloudwatch Logs

• Low ⓘ Acknowledged

Update

Client's response:

```
Skipped as low value.
```

Description: By default, CloudWatch keeps logs indefinitely, and hence is causing potentially unnecessary storage costs.

Recommendation: For every resource of type `aws_cloudwatch_log_group`, ensure that the `retention_in_days` key is set to anything else but 0 (default).

Sources:

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/cloudwatch_log_group

SEQ-ID-27 Use Builder Libraries for Special Strings

● Low ⓘ Fixed

Update
Fixed in 5d236d200270ce0f61ae57bf518f94e7bf41b490

Description: The codebase builds URLs using manual string concatenation, such as:

`/identity-instrument-master/cmd/ingress-proxy/main.go`, line 89:

```
url := "http://" + r.Host + r.URL.String()
```

This approach is error-prone and unsafe, especially when user input or variable data is embedded into URLs. Improper handling of special characters, path delimiters, or query parameters can lead to:

- Incorrect URL encoding, breaking API calls or routing logic
- Injection vulnerabilities, especially if user-controlled values are appended unchecked
- Ambiguous or malformed URLs, which can degrade reliability or security

This pattern bypasses standard encoding protections and lacks guardrails against constructing invalid or unsafe URLs — particularly problematic in systems that construct dynamic API calls, redirects, or query links.

Recommendation: Use a URL builder or parsing/encoding utility provided by the language standard library or a reputable library to construct URLs safely.

In Go, for example:

```
u := url.URL{
    Scheme: "https",
    Host:   "api.example.com",
    Path:   path.Join("foo"),
}
finalURL := u.String()
```

Benefits:

- Proper encoding of special characters
- Avoidance of trailing slash, duplicate delimiter, or injection issues
- Improved readability and maintainability

SEQ-ID-28 Signed Messages Are Prone to Replay Attacks

● Low ⓘ Fixed

Update
Fixed in 9d6695ce7508a1bef6d06b18d6e40e30d77ade65

File(s) affected: `rpc/sign.go`

Description: From the code alone, it is unclear what format messages should have. If any, they are not enforced by the `Sign` logic. This leads to an absence of checks for replay attacks, including a nonce and chainid.

Recommendation:

- Document the format of messages to be signed.
- Enforce that the chosen format is followed.
- Make sure the format includes protections against replay attacks, such as having a nonce and chainId.

SEQ-ID-29 Email-based OTP Lack of Limits May Lead to Spam and Potential DoS Attacks

● Low ⓘ Acknowledged

Update
Client's response:

```
Out of scope, won't be handled in the main repository. Instead, it will be implemented in the proxy
service, available at https://github.com/0xsequence/identity-instrument-proxy.
```

File(s) affected: `auth/otp/auth_handler.go`

Description: Malicious actors may attempt to flag 0xsequence as a spammer, simply by triggering many OTP Commit-Verify requests at once.

Unrestricted Commit-Verify OTP attempts may also lead to lower system throughput, potentially opening the door for Denial-of-Service attacks.

Recommendation:

- For any given user, restrict the number of Commit-Verify requests that can occur within a given time-window (e.g., 5 attemps per hour).
- Log all attempts higher than the accepted rate limit to flag suspicious behavior.
- Consider using CAPTCHAs before one can trigger the whole OTP email auth flow.

SEQ-ID-30
OTP-Based Email Subject May Unnecessarily Expose Users

● Low ⓘ Acknowledged

Update

Client's response:

```
Email subject and content templates are configured by Ecosystems through SequenceEcosystem Manager. As
such, it's the responsibility of each Ecosystem.
```

File(s) affected: `rpc/email/sender.go`

Description: The current email-based OTP code email is sent as part of the email subject line, which could cause the code to be leaked.

Some mobile devices, for instance, may have incoming email notifications enabled. Upon displaying a notification, the setup can be configured so that the subject line of the incoming email is displayed on the notification stack. This may happen even if the mobile device is locked.

While the 0xSequence team has taken the effort in using salt codes for both clients and servers, making it hard for one to authenticate when glancing over someone's mobile device, it is still a best practice to avoid OTP codes in the subject line. For example, OTP codes in subject lines may discourage users from actually opening the email, making them more prone to phishing attacks.

Recommendation: Remove OTP codes from email subject lines. Instead, only send them as part of the email body.

SEQ-ID-31
Docker Containers Run Their Commands as Root

● Informational ⓘ Acknowledged

Update

Client's response:

```
The Docker image in the repository is only used locally and in a (sandboxed) CI. It is neverdeployed
anywhere and is instead used as a build tool for the minimal Enclave Image Filethat the enclave
instance runs. As such, we consider this issue not applicable.
```

Fix review notes:

```
We are marking this issue as acknowledged and decreasing its severity from "High" to "Informational"
```

File(s) affected: `Dockerfile`

Description: When creating a Docker container, it is possible to set the user who is actually running the application and any command on the container. It is important to specifically use the `USER` directive in any Dockerfile to ensure that the user is not root and has unnecessary privileges.

Recommendation: Have at least one `USER` directive in your Dockerfile, and the last user directive should not reference the root user or root group.

Sources:

- https://docs.docker.com/engine/reference/builder/#user
- https://docs.docker.com/build/building/best-practices/#user

SEQ-ID-32
Serve OWASP Recommended Headers with Each Request

● Informational ⓘ Acknowledged

Update

Client's response:

```
Not applicable. The proxy located in the repository is there only for local testing purposes and is not
used in production.
```

Fix review notes:

```
We encourage any development environment to mimic the production setting as closely as possible. That
allows identifying security issues early on during the development cycle. Nonetheless, we are marking
this issue as acknowledged and decreasing its severity from "High" to "Informational".
```

File(s) affected: `cmd/ingress-proxy/main.go`

Description: The custom-built proxy (`/identity-instrument-master/cmd/ingress-proxy/main.go`) fails to set several critical HTTP response headers recommended by OWASP for web application security. These headers are intended to instruct browsers on how to behave when rendering content, limiting the attack surface for common exploits like XSS, clickjacking, MIME-sniffing, and resource abuse.

Since this is a custom proxy implementation, these protections must be explicitly coded in, unlike in production-ready reverse proxies (e.g., NGINX or Envoy) where secure defaults or plugins are available.

Without these headers, end-user browsers are left to infer behavior on their own — potentially enabling risky defaults that attackers can exploit, especially when serving dynamic or unauthenticated content.

Recommendation: Enhance your custom proxy to inject the following headers into all HTTP responses (unless explicitly overridden downstream):

```
Strict-Transport-Security: max-age=63072000; includeSubDomains; preload
X-Content-Type-Options: nosniff
X-Frame-Options: DENY
Referrer-Policy: no-referrer
Content-Security-Policy: default-src 'self'
Permissions-Policy: <MINIMUM REQUIREMENTS HERE>
Cross-Origin-Embedder-Policy: require-corp
Cross-Origin-Opener-Policy: same-origin
Cross-Origin-Resource-Policy: same-origin
```

The values of the headers above can of course be altered to the needs of the specific application.

SEQ-ID-33
Capture and Persist Operating System Logs Past the Lifetime of a Container

● Informational ⓘ Acknowledged

Update
Client's response:

```
Docker is only used for local development and as a build tool. It is not by deployed enclave instances.
```

Fix review notes:

```
We encourage any development environment to mimic the production setting as closely as possible. That
allows identifying security issues early on during the development cycle. Nonetheless, we are marking
this issue as acknowledged and decreasing its severity from "Medium" to "Informational".
```

File(s) affected: `Dockerfile`

Description: In linux systems, important operating system logs are stored in the `/var/log` subfolder. This folder should always be made available to the host through a volume, so that log tracking and log analysis systems can capture them.

Recommendation: In every Dockerfile, there should be a VOLUME directive that has `/var/log` as an argument.

Sources:
- https://docs.docker.com/engine/reference/builder/

SEQ-ID-34
Docker Compose Requires Logging Size Protection

● Informational ⓘ Acknowledged

Update

Client's response:

```
Docker is only used for local development and as a build tool. It is not by deployed enclave instances.
```

Fix review notes:

```
We encourage any development environment to mimic the production setting as closely as possible. That
allows identifying security issues early on during the development cycle. Nonetheless, we are marking
this issue as acknowledged and decreasing its severity from "Medium" to "Informational".
```

File(s) affected: `docker-compose.yml`

Description: Each Docker container is producing logs, which can be accessed via the command `docker logs <CONTAINER_ID>`. These logs grow over time, and it is recommended to limit the log file sizes in the Docker system and rotate them. The recommendation is to limit the number of files by 10, and let each file be at most 200MB big.

Recommendation: In your `docker-compose.yml` file, for each defined service, add the following:

```
logging: options: max-size: "200m" max-file: "10".
```

Sources:

- https://docs.docker.com/compose/compose-file/compose-file-v3/#logging
- https://cwe.mitre.org/data/definitions/400.html

SEQ-ID-35
Pin Versions of Packages Installed in a Docker Image

● Informational ⓘ Acknowledged

Update

Client's response:

```
Docker is only used for local development and as a build tool. It is not by deployed enclave instances.
```

Fix review notes:

```
We encourage any development environment to mimic the production setting as closely as possible. That
allows identifying security issues early on during the development cycle. Nonetheless, we are marking
this issue as acknowledged and decreasing its severity from "Medium" to "Informational".
```

File(s) affected: `Dockerfile`

Description: One desirable goal of docker images is that changing their Docker-files and re-building them in the future does not cause issues unrelated to the new changes. One way an image may not build in the future is if the Dockerfile contains instructions to install a certain program/library without pinning the version.

Recommendation: Ensure that inside a Dockerfile, you pin the versions of any installed packages using `apt`, `yum`, `apk`, `pacman`, `dnf`, `zypper`, `npm` or `pip`.

Sources:

- https://docs.docker.com/develop/develop-images/dockerfile_best-practices/#apt-get

SEQ-ID-36
Set Limited Capabilities for Containers in Docker Compose

● Informational ⓘ Acknowledged

Update

Client's response:

```
Docker is only used for local development and as a build tool. It is not by deployed enclave instances.
```

Fix review notes:

```
We encourage any development environment to mimic the production setting as closely as possible. That
allows identifying security issues early on during the development cycle. Nonetheless, we are marking
this issue as acknowledged and decreasing its severity from "Medium" to "Informational".
```

File(s) affected: `docker-compose.yml`

Description: Containers are spun up with operating system level capabilities, and the default set is most of the time above the needs of that specific container.

Recommendation: Ensure that the key `cap_drop` is set and contains at least `NET_ADMIN` and `SYS_ADMIN`.

Sources:

- https://docs.docker.com/compose/compose-file/#cap_drop

SEQ-ID-37 Do Not Mount the Host's `docker.sock` for Any Docker Container

● Informational ⓘ Acknowledged

Update

Client's response:

```
Docker is only used for local development and as a build tool. It is not by deployed enclave instances.
```

Fix review notes:

```
We encourage any development environment to mimic the production setting as closely as possible. That
allows identifying security issues early on during the development cycle. Nonetheless, we are marking
this issue as acknowledged and decreasing its severity from "Medium" to "Informational".
```

File(s) affected: `docker-compose.yml`

Description: It is possible to mount the `docker.sock` interface to a container. This enables the 'Docker within Docker' scenario, but also poses a security risk, since common Docker operations on the host are then exposed to the container.

Recommendation: Ensure that there is no volume mounted which contains the `docker.sock`.

Sources:

- https://docs.docker.com/compose/compose-file/#volumes

SEQ-ID-38 Enable Bucket Logging for S3 Buckets

● Informational ⓘ Acknowledged

Update

Client's response:

```
No S3 buckets are used in production. The S3 bucket in the dev account is only used as temporary EIF
storage, not used by the enclave.
```

Fix review notes:

```
We are marking this issue as acknowledged and decreasing its severity from "Medium" to "Informational".
```

Description: Logging access to any resources is a vital pillar for security. Hence, it is important that every S3 Bucket resource is set up with a proper logging configuration.

Recommendation: For every resource of type `aws_s3_bucket`, ensure that either a `logging` configuration block exists (deprecated), or that a resource of type `aws_s3_bucket_logging` exists referencing this bucket.

Sources:

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket
- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket_logging

SEQ-ID-39 Bind HTTP Server to a Specific Network Interface

● Informational ⓘ Acknowledged

Update
Client's response:

```
The issue is about a mock service that is only ever used in local development and never deployed.
```

Fix review notes:

```
We are marking this issue as acknowledged and decreasing its severity from "Medium" to "Informational".
```

File(s) affected: `cmd/builder-mock/main.go`

Description: The HTTP server is instantiated using `http.ListenAndServe(":9999", svc)` (`/identity-instrument-master/cmd/builder-mock/main.go` , line 11) in Go, which binds the server to all available network interfaces (`0.0.0.0`). This means the service is accessible from any network the host is connected to, including public or unintended internal networks, depending on the host's network configuration and firewall settings.

This broad exposure increases the attack surface unnecessarily and may lead to unintentional service discovery, unauthorized access attempts, or abuse if firewall rules or access controls are misconfigured or absent.

Binding to all interfaces is appropriate only in explicitly controlled environments (e.g., behind a reverse proxy or inside a secured container network), and even then, least-privilege design dictates narrowing exposure whenever possible.

Recommendation: Modify the server to bind to a specific network interface, such as localhost or a trusted internal IP.

SEQ-ID-40 Docker-Compose Should Have a Restart Policy Specified

● Informational ⓘ Acknowledged

Update
Client's response:

```
Docker is used only for local development.
```

Fix review notes:

```
We encourage any development environment to mimic the production setting as closely as possible. That
allows identifying security issues early on during the development cycle. Nonetheless, we are marking
this issue as acknowledged and decreasing its severity from "Low" to "Informational".
```

File(s) affected: `docker-compose.yml`

Description: The user should always specify the way containers are supposed to behave when an error is encountered. One mechanism is by specifying the restart policy. For each container, it is best practice to specifically set the `restart` key.

Recommendation: Ensure that the `restart` key is set to a value.

Sources:
- https://docs.docker.com/compose/compose-file/

SEQ-ID-41 Have Healthcheck Parameters in Dockerfiles

● Informational ⓘ Acknowledged

Update
Client's response:

```
Docker is used only for local development.
```

Fix review notes:

```
We are marking this issue as acknowledged and decreasing its severity from "Low" to "Informational".
```

File(s) affected: `Dockerfile`

Description: Dockerfiles have an instruction called `HEALTHCHECK`. It enables a user to define a command to figure out if the program(s) running inside the container are working properly. It is generally advisable to have healthchecks in place to assist monitoring of running containers.

Recommendation: Have at least one `HEALTHCHECK` instruction in your Dockerfile.

Sources:

- https://docs.docker.com/engine/reference/builder/#healthcheck

SEQ-ID-42 Ensure to Have a Backup Plan for Ec2 Instances

● Informational ⓘ Acknowledged

Update

Client's response:

```
The EC2 instances are ephemeral and can be shut down and restarted any time. As such, no back-ups of
EBS are required
```

Fix review notes:

```
We are marking this issue as acknowledged and decreasing its severity from "Low" to "Informational".
```

Description: When using EC2 instances, it is recommended to ensure that the attached EBS volume is being backed up regularly.

Recommendation: For every resource of type `aws_instance`, ensure that there is a resource of type `aws_backup_plan` and `aws_backup_selection`.

Sources:

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/instance

SEQ-ID-43 Enable Versioning on S3 Buckets

● Informational ⓘ Acknowledged

Update

Client's response:

```
No S3 buckets used in production.
```

Fix review notes:

```
We are marking this issue as acknowledged and decreasing its severity from "Low" to "Informational".
```

Description: S3 buckets allow for versioning of the objects stored in it, and it is generally recommended to enable it since it allows access to historical versions of objects. In older versions of Terraform, this was done inline within the bucket resource definition. This inline definition is deprecated in favor of an external resource definition. This check catches the use of the deprecated method.

Exploit Scenario:

Recommendation: For every resource of type `aws_s3_bucket`, ensure that there is no `versioning` key and if there is, replace it with a linked `aws_s3_bucket_versioning` resource linking the respective bucket.

Sources:

- https://registry.terraform.io/providers/hashicorp/aws/latest/docs/resources/s3_bucket_versioning

SEQ-ID-44 Root Port Usage Increases Attack Surface

● Informational ⓘ Acknowledged

Update

Client's response:

```
This assumes TCP. In production, the enclave is run via VSOCK.
```

Fix review notes:

```
We are marking this issue as acknowledged and decreasing its severity from "Low" to "Informational".
```

File(s) affected: `cmd/ingress-proxy/main.go`, `cmd/identity/main.go`

Description: Ports should fall within the 0-65535 range. However, both the ingress proxy and the identity instrument services do not enforce a range. The usage of non-root ports, i.e., those less than 1024, is not enforced either. This means that both binaries must be executed with root privileges, increasing the attack surface.

Recommendation: Enforce the use of non-root ports (>= 1024) to avoid the risk of privilege escalation.

SEQ-ID-45
Consider if Auth Key Maximum Ttl (90 Days) Is Needed

● Informational ⓘ Acknowledged

Update
Client's response:

```
Identity instrument is flexible and non-opinionated in the matter of auth key validity, as we can't
predict who or what will use it and their use case. The validity is thus configurable per auth key on
request. Sequence Wallet v3 uses auth keys with a validity of 5 minutes.
```

Fix review notes:

```
We are marking this issue as acknowledged and decreasing its severity from "Low" to "Informational".
```

File(s) affected: `rpc/auth.go`

Description: The current implementation allows authentication keys (`auth_key`) to have a maximum time-to-live (TTL) of up to 90 days. Long-lived credentials significantly increase the risk window for attackers: if the `auth_key` 's private key is leaked, phished, or otherwise compromised, it can be used for up to 90 days without requiring re-authentication. This undermines the security posture of the system, especially for sensitive operations or high-value accounts.

Consider if 90 days TTL is needed; otherwise, consider lowering it to a week.

Recommendation:
- Reduce the maximum TTL for authentication keys to a shorter period (e.g., 7 days) in line with industry best practices.
- For highly sensitive operations, consider even shorter TTLs (hours).
- Implement mechanisms for users to revoke keys and for the system to monitor and alert on suspicious long-lived sessions.
- If long TTLs are required for usability, ensure compensating controls are in place (e.g., device binding, anomaly detection, user notifications).

SEQ-ID-46
Non-Secp256k1 Signers Should Be Rejected Early On

● Informational ⓘ Acknowledged

Update
Client's response:

```
Non-Secp256k1 signatures are not supported right now, but they are planned, which is why they're only
checked in a single place for extensibility
```

File(s) affected: `rpc/auth.go`

Description: Non-secp256k1 signatures are not supported in the system, but are currently rejected only at the `CompleteAuth` stage.

Recommendation: Consider rejecting them early on, at the `CommitVerifier` state.

Auditor Suggestions

S1 Miscellaneous Remarks and Questions

Unresolved

Description: - `identity-instrument-master/auth/authcode/auth_handler.go`, line 100: There is a TODO item there. Is this planned to be addressed?

- `identity-instrument-master/data/signer.go`, line 159: I feel like that if-condition is not trivial to understand and would benefit from a comment.
- `identity-instrument-master/encryption/pool.go`, line 123, 143: Is the TODO fulfilled?
- `identity-instrument-master/encryption/shamir/shamir.go`: A test to keep it updated with that code should also be implemented.
- `identity-instrument-master/rpc/auth.go`, line 337: How do we know that `ecosystem` is not needing to be encoded with removing special chars with underscore?
- `identity-instrument-master/rpc/email/sender.go`, line 63: Why are they not using a string formatter here?
- `identity-instrument-master/rpc/rpc.go`, lines 78: Why are there three `test` strings in that object?

Definitions

- **High severity** – High-severity issues usually put a large number of users' sensitive information at risk, or are reasonably likely to lead to catastrophic impact for client's reputation or serious financial implications for client and users.
- **Medium severity** – Medium-severity issues tend to put a subset of users' sensitive information at risk, would be detrimental for the client's reputation if exploited, or are reasonably likely to lead to moderate financial impact.
- **Low severity** – 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 pose an immediate risk, but is relevant to security best practices or Defence in Depth.
- **Undetermined** – The impact of the issue is uncertain.
- **Fixed** – Adjusted program implementation, requirements or constraints to eliminate the risk.
- **Mitigated** – Implemented actions to minimize the impact or likelihood of the risk.
- **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).

Toolset

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

Setup

Tool Setup:

- CoGuard 1

Steps taken to run the tools:

CoGuard
n/a

Automated Analysis

CoGuard

A proprietary SAST was run on the codebase, and any identified issues have been included in the report.

Code Coverage

The test coverage review of the project indicates a strong focus on security-critical components, with robust testing of encryption, OTP flows, OpenID Connect validation, and telemetry instrumentation. These areas consistently show high coverage (80–100%), reflecting good engineering practices around cryptographic correctness and authentication workflows.

However, several important modules demonstrate only partial coverage (40–70%), particularly within the AuthCode and ID-Token handlers, metadata validation, and sections of the RPC layer. These gaps primarily concern untested error paths, boundary conditions, malformed inputs, and external dependency failures—scenarios that commonly lead to real-world issues.

A number of packages, notably CLI entrypoints and configuration loading currently show 0% coverage. While some of these files contain operational or startup logic, their lack of testing increases the risk of deployment-time regressions, misconfigurations, and unexpected runtime failures.

Overall, the project exhibits strong assurance in its core authentication and cryptographic layers but would benefit significantly from increased coverage of negative test cases, RPC boundary behavior, and configuration/initialization routines.

Coverage Data

- github.com/0xsequence/identity-instrument/auth/authcode/auth_handler.go (52.1%)
- github.com/0xsequence/identity-instrument/auth/authcode/metadata.go (73.7%)
- github.com/0xsequence/identity-instrument/auth/authcode/secrets.go (100.0%)
- github.com/0xsequence/identity-instrument/auth/idtoken/auth_handler.go (64.7%)
- github.com/0xsequence/identity-instrument/auth/idtoken/keyset.go (44.8%)
- github.com/0xsequence/identity-instrument/auth/idtoken/metadata.go (83.3%)
- github.com/0xsequence/identity-instrument/auth/idtoken/openid_config.go (72.7%)
- github.com/0xsequence/identity-instrument/auth/idtoken/validation.go (77.3%)
- github.com/0xsequence/identity-instrument/auth/otp/auth_handler.go (76.5%)
- github.com/0xsequence/identity-instrument/cmd/builder-mock/main.go (0.0%)
- github.com/0xsequence/identity-instrument/cmd/identity/main.go (0.0%)
- github.com/0xsequence/identity-instrument/cmd/ingress-proxy/main.go (0.0%)
- github.com/0xsequence/identity-instrument/config/config.go (0.0%)
- github.com/0xsequence/identity-instrument/data/auth_commitment.go (69.0%)
- github.com/0xsequence/identity-instrument/data/auth_key.go (63.5%)
- github.com/0xsequence/identity-instrument/data/cipher_key.go (68.6%)
- github.com/0xsequence/identity-instrument/data/encrypted_data.go (81.2%)
- github.com/0xsequence/identity-instrument/data/signer.go (68.5%)
- github.com/0xsequence/identity-instrument/encryption/ciphertext.go (100.0%)
- github.com/0xsequence/identity-instrument/encryption/config.go (81.0%)
- github.com/0xsequence/identity-instrument/encryption/kms/ciphertext.go (100.0%)
- github.com/0xsequence/identity-instrument/encryption/kms/remote_key.go (80.6%)
- github.com/0xsequence/identity-instrument/encryption/pool.go (83.4%)
- github.com/0xsequence/identity-instrument/encryption/shamir/shamir.go (95.9%)
- github.com/0xsequence/identity-instrument/o11y/enclave_provider.go (63.2%)
- github.com/0xsequence/identity-instrument/o11y/http_client.go (76.5%)
- github.com/0xsequence/identity-instrument/o11y/logger.go (100.0%)
- github.com/0xsequence/identity-instrument/o11y/middleware.go (84.2%)
- github.com/0xsequence/identity-instrument/o11y/trace.go (93.2%)
- github.com/0xsequence/identity-instrument/o11y/traced_auth_handler.go (100.0%)
- github.com/0xsequence/identity-instrument/o11y/traced_cache.go (52.4%)
- github.com/0xsequence/identity-instrument/o11y/traced_rpc.go (0.0%)
- github.com/0xsequence/identity-instrument/proto/admin/admin.gen.go (39.9%)
- github.com/0xsequence/identity-instrument/proto/auth_commitment.go (100.0%)
- github.com/0xsequence/identity-instrument/proto/auth_id.go (100.0%)
- github.com/0xsequence/identity-instrument/proto/builder/authenticated_client.go (86.4%)
- github.com/0xsequence/identity-instrument/proto/builder/builder.gen.go (34.1%)
- github.com/0xsequence/identity-instrument/proto/builder/mock.go (100.0%)
- github.com/0xsequence/identity-instrument/proto/clients/identity-instrument.gen.go (44.6%)
- github.com/0xsequence/identity-instrument/proto/clients/types.go (100.0%)
- github.com/0xsequence/identity-instrument/proto/identity-instrument.gen.go (39.2%)
- github.com/0xsequence/identity-instrument/proto/identity.go (97.4%)
- github.com/0xsequence/identity-instrument/proto/key.go (69.2%)
- github.com/0xsequence/identity-instrument/proto/params.go (96.2%)
- github.com/0xsequence/identity-instrument/proto/scope.go (92.9%)
- github.com/0xsequence/identity-instrument/proto/signer_data.go (75.0%)
- github.com/0xsequence/identity-instrument/rpc/admin.go (69.6%)
- github.com/0xsequence/identity-instrument/rpc/auth.go (64.8%)
- github.com/0xsequence/identity-instrument/rpc/awscreds/provider.go (73.9%)
- github.com/0xsequence/identity-instrument/rpc/awscreds/static_provider.go (66.7%)
- github.com/0xsequence/identity-instrument/rpc/email/format.go (100.0%)
- github.com/0xsequence/identity-instrument/rpc/email/sender.go (72.1%)
- github.com/0xsequence/identity-instrument/rpc/internal/attestation/context.go (100.0%)
- github.com/0xsequence/identity-instrument/rpc/internal/attestation/middleware.go (80.3%)
- github.com/0xsequence/identity-instrument/rpc/internal/attestation/userdata.go (93.8%)
- github.com/0xsequence/identity-instrument/rpc/internal/signature/ethereum.go (84.6%)
- github.com/0xsequence/identity-instrument/rpc/internal/signature/middleware.go (71.4%)
- github.com/0xsequence/identity-instrument/rpc/internal/signature/webcrypto.go (93.8%)
- github.com/0xsequence/identity-instrument/rpc/rpc.go (57.9%)
- github.com/0xsequence/identity-instrument/rpc/sign.go (60.3%)

Changelog

- 2025-07-29 - Initial report
- 2025-12-02 - Final report

About Quantstamp

Quantstamp is a global leader in blockchain security. Founded in 2017, Quantstamp's mission is to securely onboard the next billion users to Web3 through its best-in-class Web3 security products and services.

Quantstamp's team consists of cybersecurity experts hailing from globally recognized organizations including Microsoft, AWS, BMW, Meta, and the Ethereum Foundation. Quantstamp engineers hold PhDs or advanced computer science degrees, with decades of combined experience in formal verification, static analysis, blockchain audits, penetration testing, and original leading-edge research.

To date, Quantstamp has performed more than 500 audits and secured over $200 billion in digital asset risk from hackers. Quantstamp has worked with a diverse range of customers, including startups, category leaders and financial institutions. Brands that Quantstamp has worked with include Ethereum 2.0, Binance, Visa, PayPal, Polygon, Avalanche, Curve, Solana, Compound, Lido, MakerDAO, Arbitrum, OpenSea and the World Economic Forum.

Quantstamp's collaborations and partnerships showcase our commitment to world-class research, development and security. We're honored to work with some of the top names in the industry and proud to secure the future of web3.

Notable Collaborations & Customers:

- Blockchains: Ethereum 2.0, Near, Flow, Avalanche, Solana, Cardano, Binance Smart Chain, Hedera Hashgraph, Tezos
- DeFi: Curve, Compound, Maker, Lido, Polygon, Arbitrum, SushiSwap
- NFT: OpenSea, Parallel, Dapper Labs, Decentraland, Sandbox, Axie Infinity, Illuvium, NBA Top Shot, Zora
- Academic institutions: National University of Singapore, MIT

Quantstamp