Quantstamp

Beep

Executive Summary

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

Type	Infra
Timeline	2025-11-25 through 2025-11-25
Language	TypeScript
Methods	Architecture Review, Unit Testing, Functional Testing, Computer-Aided Verification, Manual Review
Specification	None
Source Code	• https://github.com/beep-it/beep-server #7ff3bd6 • beep-it/beep-frontend #654879b • beep-it/beep-sdk #39cf81c • beep-it/infra-terraform #f17706c
Auditors	• Albert Heinle Auditing Engineer • Tianyi Hu Auditing Engineer

Documentation quality	Undetermined
Test quality	Undetermined
Total Findings	52 Unresolved: 13 Fixed: 31 Acknowledged: 6 Mitigated: 2
High severity findings	24 Fixed: 20 Acknowledged: 3 Mitigated: 1
Medium severity findings	15 Fixed: 11 Acknowledged: 3 Mitigated: 1
Low severity findings	12 Unresolved: 12
Undetermined severity findings	0
Informational findings	1 Unresolved: 1

Summary of Findings

Our assessment identified some systemic gaps across the application, its use of large-language-model (LLM) workflows, and the surrounding cloud infrastructure. The most critical issues center on input and output handling for LLM interactions. The current codebase does not implement any form of query-sanitization middleware before user prompts are sent to LLMs, nor are the LLM responses sanitized before being consumed by downstream systems. In one notable instance, an LLM-generated string is used directly to construct a SQL query. This pattern creates an exposure window for prompt-based injection attacks that could bypass ORM protections and lead to arbitrary query execution. With very little effort of prompt generation injection, we were able to make the LLM perform non-app specific tasks and escape from the merchant-specific restrictions.

The cloud environment was showing several default configurations which are recommended to be changed in a production environment. Many default Google Cloud Platform (GCP) configurations remain unchanged, leaving the deployment aligned more with general-purpose defaults than with hardened production posture. Adjusting these defaults, particularly around IAM scoping, network boundaries, and logging settings, would significantly strengthen the environment's security baseline.

Within the application itself, we identified several cryptographic and access-control concerns. Passwords and other sensitive values are still hashed using legacy algorithms such as BCrypt that no longer meet current best-practice recommendations for iteration strength. Additionally, API keys used within the system do not appear to have built-in expiry or rotation mechanisms, increasing the risk that compromised credentials could remain valid indefinitely.

Resource-consumption controls also require improvement. Current safeguards do not adequately mitigate API wallet–draining attacks or other scenarios in which an adversary could drive excessive LLM usage and incur material cost. While the ORM provides solid protection against most injection vectors, some isolated but high-impact code paths bypass the ORM entirely, allowing raw queries and introducing unnecessary injection risk.

Finally, we observed a user-controlled redirect parameter that is accepted and executed without adequate validation. This creates the potential for open-redirect attacks, which can be leveraged for phishing, credential interception, or other downstream compromise.

Overall, remediating these issues will require tightening LLM data-flow controls, modernizing security mechanisms for authentication and key management, enforcing consistent ORM usage, and hardening both cloud and application-level configurations. Addressing these gaps will materially improve the platform's security posture and reduce exposure to high-impact attack vectors.

AI GENERATED FIX REVIEW SUMMARY:

In the latest code review, vulnerabilities QSP-1, QSP-2, QSP-4, QSP-5, QSP-6, QSP-7, QSP-8, QSP-9, QSP-10, QSP-12, QSP-18, QSP-19, QSP-20, QSP-21, QSP-22, QSP-24, QSP-26, QSP-27, QSP-28, QSP-30, and QSP-39 have all been fixed, with specific explanations provided for each resolution, such as input sanitation, improved hashing algorithms, updates to packages, and enhanced security mechanisms. However, vulnerabilities QSP-3, QSP-11, QSP-13, QSP-14, QSP-15, QSP-16, QSP-31, QSP-32, QSP-33, QSP-34, QSP-35, QSP-36, and several others (QSP-25, QSP-29, QSP-38, QSP-40, QSP-41, QSP-42, QSP-43, QSP-44, QSP-45, QSP-46, QSP-47, QSP-48, QSP-49, QSP-50, QSP-51, and QSP-52) were acknowledged as either low priority or accepted risks, resulting in them remaining unresolved. The client appears to prioritize fixes based on perceived risk levels, which may merit further discussion to ensure a comprehensive approach to security vulnerabilities.

ID	DESCRIPTION	SEVERITY	STATUS
BEE-1	Unsanitized User Input	● High	Fixed
BEE-2	Insecure SQL Query Execution with Unsanitized Input coming from LLM output	● High	Mitigated
BEE-3	Unhandled Resource Consumption	● High	Acknowledged
BEE-4	Use of legacy Hashing Algorithm	● High	Fixed
BEE-5	High and Critical CVEs present in code-base	● High	Fixed
BEE-6	Specify permissions for each GitHub Action	● High	Fixed
BEE-7	Do not run Docker images as root	● High	Fixed
BEE-8	Insufficient Sanitization of Sensitive Information in Application Logs	● High	Fixed
BEE-9	Unvalidated User-Supplied URL Used for HTTP Redirect	● High	Fixed
BEE-10	Content Security Policy (CSP) Disabled in Express.js Helmet Configuration	● High	Fixed
BEE-11	Enable Transit Encryption for Google Cloud Redis	● High	Fixed
BEE-12	Enable Google Cloud Redis Authentication	● High	Fixed
BEE-13	Restrict Service Account Permissions	● High	Fixed
BEE-14	Enforce Shielded GCP Compute Instances	● High	Fixed
BEE-15	Disable Project-Wide SSH Keys for GCP Instances	● High	Fixed
BEE-16	Restrict IP Forwarding on GCP Compute Instances	● High	Acknowledged
BEE-17	Block RDP Access to GCP Compute Instances	● High	Fixed
BEE-18	Unlimited Message History in Chat Context	● High	Fixed

ID	DESCRIPTION	SEVERITY	STATUS
BEE-19	Potential URL Injection Vulnerability	● High	Fixed
BEE-20	API Key Validation Lacks Expiration Check	● High	Fixed
BEE-21	Insecure Rate Limiting Configuration	● High	Fixed
BEE-22	Lack of Secure Random Number Generation	● High	Fixed
BEE-23	Insufficient per-user limitation of LLM queries	● High	Acknowledged
BEE-24	Insecure OAuth Implementation	● High	Fixed
BEE-25	Match local Redis security settings to Production	● Medium	Fixed
BEE-26	Match local Postgres security settings to Production	● Medium	Fixed
BEE-27	Pin actions to SHA sum to prevent supply chain attacks	● Medium	Fixed
BEE-28	Logical code inconsistencies	● Medium	Fixed
BEE-29	Presence of Unresolved TODO Comments in Codebase	● Medium	Mitigated
BEE-30	Risk of Reverse Tabnapping Due to Use of target="_blank" Without rel Attributes	● Medium	Fixed
BEE-31	Drift from use of ORM	● Medium	Acknowledged
BEE-32	Disable Default GCP Network	● Medium	Fixed
BEE-33	Enable DNSSEC with Modern Signing Key for GCP DNS Managed Zone	● Medium	Acknowledged
BEE-34	Restrict GCP Firewall Source Ranges	● Medium	Acknowledged
BEE-35	Ensure GCP Project Audit Logging Enabled	● Medium	Fixed
BEE-36	Enable GCP Container Image Vulnerability Scanning	● Medium	Fixed
BEE-37	Input Length Restrictions Lacking	● Medium	Fixed
BEE-38	Lack of Pagination in Queries	● Medium	Fixed
BEE-39	Sensitive Implementation Details Exposure	● Medium	Fixed
BEE-40	Persist operating system logs in Docker images	● Low	Unresolved
BEE-41	Enable healthchecks for all Docker images	● Low	Unresolved
BEE-42	Unsafe Temporary File Creation Without Using a Proper Tempfile Library	● Low	Unresolved
BEE-43	Enable Compute Subnetwork Flow Logs	● Low	Unresolved
BEE-44	Restrict Public IP Access to GCP Compute Instances	● Low	Unresolved
BEE-45	Header Leaks Implementation Details	● Low	Unresolved

ID	DESCRIPTION	SEVERITY	STATUS
BEE-46	Inefficient Multi-Layer Docker Build	• Low	Unresolved
BEE-47	Duplicated Conditional Logic in ChatContext	• Low	Unresolved
BEE-48	Centralize Type Definitions	• Low	Unresolved
BEE-49	Use of Integer Type for Unique Identifiers	• Low	Unresolved
BEE-50	Inefficient Delay in Treasury Withdrawal Service	• Low	Unresolved
BEE-51	Chat history stored and communicated through URL query parameters	• Low	Unresolved
BEE-52	Define a timeout for any GitHub Actions	• Informational	Unresolved

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):

- Denial of service / logical oversights
- Access control
- Key rotation issues
- Insecure storage of credentials
- Downtime
- Infrastructure misconfiguration
- Privilege escalation
- Slashing risk due to inadequate resource isolation
- General cybersecurity risks according to NIST 800-53
- LLM related risks

Methodology

- Review of all configuration files in your infrastructure, coming from both code repositories and direct API connections to the cloud provider.
- Apply CIS, DoD STIG and other applicable benchmarks to discovered configurations.
- Review Architectural choices and determine if all required infrastructure components are present.
- Review Code for CWEs.
- Analyze the code and infrastructure against OWASP top 10 and OWASP LLM top 10.
- Analyze supply chain security and flag CVEs and other vulnerability types.

Scope

Files Included

Repo: https://github.com/beep-it/beep-server(7ff3bd6a68475cdf76aec987ce9672f8f6d825bd) Files: / Repo: https://github.com/beep-it/beep-frontend(654879b7050b15240a2aadfdfd55a8ebd7392369) Files: /*/ Repo: https://github.com/beep-it/beep-sdk(39cf81cb74ef2b700e55e15d7dd4365faf1506cd) Files: */* Repo: https://github.com/beep-it/infra-terraform(f17706cfa58346111865fd32356d55c2e795f9df)

Findings

BEE-1 Unsanitized User Input

• High Fixed

Update

Sanitization measures have been implemented, and the boundary is now properly enforced and non-predictable through the introduction of a random boundary parameter within the tag, along with the replacement of special characters in user-supplied input. As a result, the originally identified queries no longer function, as expected.

However, the current sanitization approach relies primarily on regular-expression-based filtering. While this mitigates a broad range of attack attempts, it may not comprehensively address all edge cases. It is therefore recommended to continuously enhance the filtering logic and, where feasible, transition to a well-maintained, officially supported library. This would allow the organization to benefit from community-vetted security expertise and ongoing updates.

Update

Marked as "Fixed" by the client.
Addressed in: `2ff50d43cbf68b5f3438a85832f7add2e7545237`.
The client provided the following explanation:

> added input sanitation

Description: User input in the `query` parameter is not properly sanitized, allowing potential injection of malicious code. Even though structured prompts are used as a mitigation, the early escape of `</user-query>` is not caught, leaving the system vulnerable. This issue affects multiple files, including `beep-server-dev/src/agents/money-talks.agent.ts`, `beep-server-dev/src/services/dynamic-query/prompts.ts`, and `beep-server-dev/src/services/user-memory.service.ts` (specifically the `updateUserMemory` function).

Furthermore, the LLM is currently reacting to non-app-specific queries. We managed to get it to do the following:

- Create a Java Hello-world program
- Create an ASCII version of a drawing of a 5-year old.
- Output of the total revenue across all merchants.
- Total number of merchants.
- All merchants on the platform and their revenue.
- Enumeration of some wallet addresses.
- Retrieving all invoices by merchants, and the total amount.
- Retrieving the total of all treasury balances across all merchant wallets.
- Disclosure what actions Money Talks can perform and what data it has access to.
- Confused Deputy attack: Asking to delete all products of the merchant "Beep Demo" was responded with the fact that there was nothing in the catalogue of "Beep Demo", instead of refusal.

While it is difficult to contain the functionality of an LLM, one should at least have safeguards in place, sufficiently ensuring that user input is domain-specific to the app. Furthermore, a quota of allowed LLM calls per user should be enforced to avoid large API token bills.

Recommendation: 1. Implement proper input validation and sanitization for the `query` parameter and any other user input fields.
2. Use secure methods for handling user input, such as input whitelisting, escaping special characters, or using prepared statements (if applicable).
3. Ensure that all potential injection vectors, including early escapes and edge cases, are properly handled.
4. Consider using a trusted third-party library or framework for input validation and sanitization to reduce the risk of implementation errors.
5. Implement strict input validation policies and regularly update them based on emerging threats and best practices.
6. Implement a time-bound quota of queries a user can send to the LLM.

BEE-2

Insecure SQL Query Execution with Unsanitized Input coming from LLM output

● High ⓘ Mitigated

Update

Additional query validation has been implemented through the utils/sql-security.util.ts helper module, providing deterministic checks intended to mitigate multiple layers of risk.

However, the filtering logic for disallowed operations currently assumes that the relevant SQL keyword appears at the beginning of the query (i.e., via startsWith checks). SQL queries may also begin with non-operative constructs such as WITH (...) AS x, , which could be leveraged to embed meaningful operations later in the statement. In such cases, it may be possible to satisfy most existing regular-expression checks by placing matching patterns within a non-operative WITH clause.

While exploiting this behavior would require additional effort and complexity, and therefore increases the attacker's burden, the issue is considered mitigated for the purposes of this re-audit. Nevertheless, it is recommended to explicitly account for this query structure in future filtering enhancements to further reduce residual risk.

Update

> Marked as "Fixed" by the client.
> Addressed in: `2ff50d43cbf68b5f3438a85832f7add2e7545237` .
> The client provided the following explanation:
>
> > added input sanitation

Description: The application is directly executing SQL queries derived from unsanitized LLM output, which poses a significant risk of SQL injection attacks. Specifically, in the file `beep-server-dev/src/services/dynamic-query/dynamic-query.service.ts` at line 39, the LLM output is used to construct a SQL query without proper input validation or sanitization.

Recommendation: 1. Implement input validation and sanitization measures for all user-supplied data, including LLM output, before using it to execute SQL queries.
2. Use parameterized queries or prepared statements to separate the query logic from the user input, preventing the injection of malicious code.
3. Implement strict input validation rules and whitelist allowed characters or patterns for any user input used in SQL queries.
4. Regularly review and update the input validation and sanitization mechanisms to address emerging threats and vulnerabilities.

BEE-3 Unhandled Resource Consumption

● High ⓘ Acknowledged

> **Update**
> Marked as "Acknowledged" by the client.
> The client provided the following explanation:
>
> > I looked at this finding and these are my findings:
> > The code execution happens in e2b, not in Beep, so there is no risk of resource usage exhaustion on our end, we have a sandbox template with reserved resources on E2B and that should do it for the requirements we have, which is drawing a single chart.
> > Additionally, we’ve set up a semaphore mechanism that limits the number of parallel executions to 20 (the limit of our current plan in e2b), so no overloading should happen for now - we could switch to paid plan and increase the limits if necessary
> > The sandbox we use there is set up based on our requirements for generating charts
> > We are the only ones that execute code in those sandboxes and there are several layers that transform a user query to an executable code, so there is no option for the user to run custom code there - we run it and the code would always attempt to draw a chart based on data provided by our agent, not directly from the user and nothing else.
> > We are logging every step of the way in details and we also have performance monitoring where we log the execution time of each run.
> > So, based on this report, I consider this as a non-issue.

Description: The `code-execution.service.ts` file in the `beep-server-dev/src/services/code-execution` directory does not implement any measures to monitor or limit CPU and memory usage during code execution. This could potentially lead to resource exhaustion attacks, where an attacker could submit malicious code that consumes excessive system resources, causing denial of service or system instability.

Recommendation: 1. Implement resource monitoring and limiting mechanisms in the `code-execution.service.ts` file.
2. Set reasonable limits on CPU and memory usage for each code execution instance.
3. Log and alert on resource usage violations and failed code executions.
4. Consider implementing a queuing system or rate-limiting to prevent overloading the code execution service.

BEE-4 Use of legacy Hashing Algorithm

● High ⓘ Fixed

> **Update**
> The client utilizes Argon2 for password hashing and supports a fallback to BCrypt for the verification of legacy hashes. This is a reasonable and industry-accepted approach for phased migration.

> **Update**
> Marked as "Fixed" by the client.
> Addressed in: `fc25ce62001eac8b5cb8a598ca41037cecbe7f86` .
> The client provided the following explanation:
>
> > Using Argon2 hashing algo

Description: The application is using the Bcrypt hashing algorithm for storing sensitive data such as API keys (`beep/beep-server-dev/src/services/apiKey.service.ts`). Bcrypt is an older hashing function that is no longer considered secure against modern attacks, particularly those leveraging powerful hardware like GPUs and ASICs. The use of a weak hashing algorithm increases the risk of compromised data in the event of a security breach.

Recommendation: Replace the use of Bcrypt with a more secure and up-to-date hashing algorithm such as Argon2, which is designed to be resistant to GPU and ASIC-based attacks. Ensure that the new hashing algorithm is configured with appropriate parameters (e.g., memory cost, iterations) to provide the desired level of security. Additionally, consider implementing a key derivation function (KDF) to further strengthen the security of sensitive data storage.

BEE-5 High and Critical CVEs present in code-base

● High ⓘ Fixed

Update
Dependabot was configured in the repositories, and the CVEs have been addressed or mitigated.

Update
Marked as "Fixed" by the client.
Addressed in: `3564f7210d2d1ee912e7e77733da085d2b7c2cc5`, `198f1489a46a975f706324cdb50a4e14c8f40c18`, `bbe0f53a4cb6c8e0296d675c9fa80c63aa3b377a`.
The client provided the following explanation:

> Fixed and updated respective node packages

Description: The following CVEs were discovered across the shared repositories:

- CVE-2025-64756: This affects the packages pkg:npm/glob@10.4.5, pkg:npm/glob@11.0.3. Directly linked CWEs: 78.
- CVE-2022-39353: This affects the packages pkg:npm/xmldom@0.1.31. Directly linked CWEs: 20, 1288.
- CVE-2025-12735: This affects the packages pkg:npm/expr-eval@2.0.2.
- CVE-2025-13204: This affects the packages pkg:npm/expr-eval@2.0.2. Directly linked CWEs: 1321.
- CVE-2025-3194: This affects the packages pkg:npm/bigint-buffer@1.1.5. Directly linked CWEs: 120.
- CVE-2025-59288: This affects the packages pkg:npm/playwright@1.54.2. Directly linked CWEs: 347.
- CVE-2025-64756: This affects the packages pkg:npm/glob@10.4.5. Directly linked CWEs: 78.
- CVE-2024-39338: This affects the packages pkg:npm/axios@1.7.2. Directly linked CWEs: 918.
- CVE-2025-27152: This affects the packages pkg:npm/axios@1.7.2. Directly linked CWEs: 918.
- CVE-2025-3194: This affects the packages pkg:npm/bigint-buffer@1.1.5. Directly linked CWEs: 120.
- CVE-2025-58754: This affects the packages pkg:npm/axios@1.7.2, pkg:npm/axios@1.9.0. Directly linked CWEs: 770.
- CVE-2025-3194: This affects the packages pkg:npm/bigint-buffer@1.1.5. Directly linked CWEs: 120.
- CVE-2025-57822: This affects the packages pkg:npm/next@15.4.4. Directly linked CWEs: 918.
- CVE-2025-58754: This affects the packages pkg:npm/axios@1.9.0. Directly linked CWEs: 770.
- CVE-2025-64756: This affects the packages pkg:npm/glob@10.4.5, pkg:npm/glob@11.0.3. Directly linked CWEs: 78.

Recommendation: Ensure that all of the project's supply chain dependencies are kept up to date with the latest version, and make CVE checks part of the PR process.

BEE-6 Specify permissions for each GitHub Action

● High ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `01bc2a8be7d8a03f44f36f507051554b9974f4c6`, `c0c343c2d0f23252dbf70dccfdccbde862c083fb`.
The client provided the following explanation:

> updated PR yaml

Description: The default GitHub token used for GitHub actions has permissions to do a lot of things. These should be limited depending on the specific workflow.

Recommendation: For every workflow, ensure that the `permission` section exists and only the necessary permissions are set.

BEE-7 Do not run Docker images as root

● High ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `35e1d92970726a4497239adb24327cb5b94addd1`.
The client provided the following explanation:

> updated docker file

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.

Affected Files:

- `./beep-server-dev/Dockerfile`

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

BEE-8 Insufficient Sanitization of Sensitive Information in Application Logs

● High ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `4cbed16b04ffafe624974cbfbd260f7df718a09b`.
The client provided the following explanation:

> using deep redact to reduce what is logged

Description: The application's logging mechanism does not consistently scrub or redact sensitive data before writing entries to log files. As a result, logs may inadvertently contain information such as authentication tokens, passwords, API keys, session identifiers, personal data, or other security-relevant values. Storing sensitive data in logs increases the blast radius of a breach, as logs are often widely accessible to operational teams, backup systems, centralized logging platforms, or third-party monitoring tools. Logs typically have long retention periods, making accidental exposure persistent and difficult to fully remediate. Failure to sanitize logs can also violate compliance requirements that mandate minimization of sensitive data exposure and strict controls over data storage practices.

Affected Files:

- `beep-server-dev/src/routes/auth.routes.ts`, line 69: You should not expose sensitive information like session ids or all cookies unscrubbed potentially.
- `beep/beep-server-dev/src/mcp/http.ts`, line 119: insufficient sanitization of potential sensitive information in logs.

Recommendation:

- Identify all sensitive fields (credentials, tokens, secrets, PII, etc.) and implement a logging policy mandating their redaction.
- Update logging functions and middleware to automatically sanitize sensitive values before writing log entries.
- Replace ad-hoc console.log or free-form logging with structured logging libraries that support field filtering and redaction.
- Conduct a repository-wide scan to locate any remaining direct logs of sensitive data and remove or sanitize them.
- Add automated linting or CI checks to detect logging of known sensitive patterns.
- Review log storage locations, access permissions, and retention policies to ensure compliance with security and privacy standards.

BEE-9 Unvalidated User-Supplied URL Used for HTTP Redirect

● High ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `3608dc11e0bcb5469682a7cf95967cc35ac2de67`.
The client provided the following explanation:

> adding validator function on redirect url

Description: The application performs an HTTP redirect using a value that is directly supplied by the user without proper validation, sanitization, or allowlisting. This introduces an open redirect vulnerability, which allows an attacker to craft URLs that appear to originate from the legitimate application but redirect users to malicious external destinations. Open redirects are commonly used in phishing, credential-harvesting, and malware distribution campaigns because they leverage the reputation of a trusted domain to increase click-through likelihood. In some scenarios, they can also be chained with other vulnerabilities to escalate impact, such as bypassing authentication workflows or facilitating OAuth token hijacking.

Affected Files:

- `beep-server-dev/src/routes/auth.routes.ts`, route `/auth/logout`, lines 310 - 333: `redirectTo` is assigned from the query string of the request, and without further checking or sanitization used as `redirect` instruction at the end of the function.
- `beep-server-dev/src/routes/auth.routes.ts`, lines 25-63: `redirectTo` can be assigned from the query string of the request, and without further checking or sanitization used as `redirect` instruction at the end of the function.

Recommendation:

- Validate the redirect target against a strict allowlist of permitted internal paths or trusted external domains.
- When redirecting within the application, enforce relative URLs rather than absolute user-supplied URLs.
- Reject or sanitize any input containing dangerous schemes (javascript:, data:, vbscript:) or external hosts not on the allowlist.
- Implement centralized redirect helper functions to ensure consistency and prevent reintroduction of insecure patterns.
- Add unit and integration tests covering expected valid redirects and rejection of malicious inputs.

BEE-10 Content Security Policy (CSP) Disabled in Express.js Helmet Configuration

● High ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `c4e9bb5adcedd8a119029043855e3ed4ee15dfee`.
The client provided the following explanation:

> provided explicit content security policies

Description: The Express.js application uses Helmet for HTTP security headers, but the Content Security Policy (CSP) module is explicitly disabled. CSP is a critical defense-in-depth control that restricts which external resources (scripts, styles, iframes, images) the browser may load. Without an enforced CSP, the application is significantly more susceptible to cross-site scripting (XSS), clickjacking, data exfiltration via injected scripts, and other browser-based attacks. Disabling CSP leaves the application dependent solely on input sanitization and other reactive controls, reducing overall resiliency against injection vulnerabilities. Modern applications, especially those handling sensitive data, are expected to implement a tailored CSP that aligns with their front-end architecture. The absence of CSP represents a meaningful degradation of client-side security posture.

Affected files:

`beep-server-dev/src/server.ts`, line 111

Recommendation:

- Re-enable Helmet's contentSecurityPolicy middleware in the Express.js configuration.
- Start with a restrictive default policy (e.g., default-src 'self') and iteratively refine it to allow only the specific external domains required by the application (e.g., CDNs, analytics endpoints).
- Avoid broad or unsafe directives such as unsafe-inline, unsafe-eval, or wildcard (*) sources. Use nonces or hashes for inline scripts when necessary.
- Validate the final policy using browser developer tools and automated scanners (e.g., Mozilla Observatory, securityheaders.com).
- Implement CSP violation reporting (report-uri or report-to) to detect and monitor attempted policy violations in production.
- Establish a secure-coding guideline ensuring that CSP remains enabled and updated as part of ongoing development and deployment processes.

BEE-11 Enable Transit Encryption for Google Cloud Redis

● High ⓘ Fixed

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

> Updated GCP:
> transit_encryption_mode = "SERVER_AUTHENTICATION"

Description: Google Cloud Redis service does not have encryption in transit enabled by default.

Recommendation: For every resource of type `google_redis_instance`, ensure that `transit_encryption_mode` is not set to `DISABLED` (default).

BEE-12 Enable Google Cloud Redis Authentication

● High ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `1d7b60bc6d4660fa10700700a7b056e5e8788b1e`.
The client provided the following explanation:

> enable redis url

Description: Google Cloud Redis service does not have authentication enabled by default.

Recommendation: For every resource of type `google_redis_instance`, ensure that `auth_enabled` is set to `true` (default is `false`).

BEE-13 Restrict Service Account Permissions

● High ⓘ Fixed

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

> Updated restriction in GCP

Description: Service accounts are created to have particular, application specific permission. Granting them the role of an owner is not recommended.

Recommendation: For every resource of type `google_project_iam_binding`, `google_project_iam_member`, `google_organization_iam_member` and `google_organization_iam_binding` ensure that if the role is `owner`, no members matching `.*.gserviceaccount.com` are granted this role.

BEE-14 Enforce Shielded GCP Compute Instances

● High ⓘ Fixed

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

> added this commit: f17706cfa58346111865fd32356d55c2e795f9df
> to new repo: https://github.com/beep-it/infra-terraform

Description: Any compute instance on Google Cloud can be configured to be shielded, which adds an extra layer of security against malware and rootkits provided by Google. Any instance should generally be configured to be shielded.

Recommendation: For every resource of type `google_compute_instance`, ensure that the block `shielded_instance_config` exists, and that the sub-keys `enable_secure_boot`, `enable_vtpm` and `enable_integrity_monitoring` are each set to true (some default to true).

BEE-15 Disable Project-Wide SSH Keys for GCP Instances

● High ⓘ Fixed

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

> Updated in GCP

Description: We recommend to use separate, managed, SSH keys for each Google Cloud compute instance. Hence, the feature to use project-wide SSH keys should be disabled for each compute instance.

Affected Resources:

- `vpn-prod`

Recommendation: For every resource of type `google_compute_instance`, ensure that the block `metadata` contains the key-value pair `block-project-ssh-keys` to be `true`.

BEE-16 Restrict IP Forwarding on GCP Compute Instances

● High ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

> vpn-prod is a VPN relay instance.
> In this case, and only for this instance, we need ip_forward enabled.
> We are aware of this, so this is an accepted risk.

Description: Enabling a compute instance to do ip forwarding is generally not needed in Google cloud, and poses a network sniffing risk to the traffic in your virtual private network.

Affected Files:
- `vpn-prod`

Recommendation: For every resource of type `google_compute_instance`, ensure that the key `can_ip_forward` is `false` (default).

BEE-17 Block RDP Access to GCP Compute Instances

● High ⓘ Fixed

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

> Updated in GCP

Description: By default, the Google compute instances allow traffic coming from port 3389 (remote desktop protocol). The RDP protocol has caused many security incidences in the past, and its use should be avoided.

Recommendation: For every resource of type `google_compute_firewall`, ensure that there is at least one `deny` block with `protocol` set to `tcp`, and `ports` containing `3389`.

BEE-18 Unlimited Message History in Chat Context

● High ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `6a0037c7821001fc2c296273786ad650619959ce`.
The client provided the following explanation:

> adding length restriction to chat history messages

Description: The `ChatContext.tsx` file in the `beep-frontend/src/app/context` directory does not impose a limit on the length of the previous messages stored in the chat history. This can potentially be exploited for a wallet-draining attack, as tokens can accumulate through user inputs over an extended period.

Recommendation:
- Implement a maximum length for the chat history array, limiting the number of previous messages stored.
- Periodically clear the chat history or remove older messages when the maximum length is reached.
- Sanitize and validate user inputs before adding them to the chat history to prevent potential token accumulation.

BEE-19 Potential URL Injection Vulnerability

● High ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `eac6c37c925e0ddfb0637171af8dbabdaee50fcc`.
The client provided the following explanation:

> added url sanitation

Description: The `buildApiUrl` function in the `beep-frontend/src/config/api.ts` file lacks proper input sanitization and validation. It allows arbitrary strings to be used for generating URLs, which could potentially lead to Server-Side Request Forgery (SSRF) or other injection attacks. This represents a significant security risk.

Recommendation: 1. Implement strict input validation and sanitization for the `buildApiUrl` function to ensure that only trusted and expected values are accepted.

2. Use a secure URL building library or utility function that performs necessary encoding and validation to prevent injection attacks.
3. Consider implementing a whitelist of allowed URL components or patterns to restrict the possible inputs for the `buildApiUrl` function.
4. Regularly review and update the input validation and sanitization mechanisms to address newly discovered vulnerabilities or attack vectors.

BEE-20 API Key Validation Lacks Expiration Check

● High ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `85d8d13e4d70ee4e137056b73135334073cdc9fc`, `34c53ae0226e3c6740505b98a40800c570baa7c4`, `fc25ce62001eac8b5cb8a598ca41037cecbe7f86`.
The client provided the following explanation:

> Added expiration dates to API keys

Description: The API key validation process in the codebase does not include checks for expiration or time-limited validity. This could potentially allow compromised or revoked API keys to continue being used indefinitely, posing a security risk.

Recommendation: 1. Implement a mechanism to associate API keys with an expiration date or validity period.

2. Modify the `validateApiKey` function in `beep-server-dev/src/services/apiKey.service.ts` to check the expiration date/validity period of the API key during the validation process.
3. Establish a process to periodically rotate or refresh API keys to mitigate the risk of long-lived API keys being compromised.

BEE-21 Insecure Rate Limiting Configuration

● High ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `8324a722d4ac5bc7e193095e91e4723a2a8a6f1e`.
The client provided the following explanation:

> The fix was two part, we adjusted the in application rate limiting as well as manage rate limiting via load balancers on GCP

Description: The rate limiting mechanism in place appears to be configured insecurely, allowing an excessive number of requests (100,000 as per `rateLimit.middleware.ts`) per IP address. This could potentially enable attackers to overload the system with a relatively small number of IP addresses, leading to a Denial of Service (DoS) condition.

Recommendation:

- Review the rate limiting requirements and adjust the configuration to a more reasonable limit, such as 100 requests per minute per IP address. ely relying on IP addresses.
- Implement mechanisms to detect and block abusive behavior, such as temporarily blocking IP addresses that exceed the rate limit.
- Regularly monitor and review the rate limiting configuration and adjust it as necessary based on the observed traffic patterns and potential threats.
- The `express-rate-limit` package, which is employed, does not come with functionality to allow for short bursts. We recommend to maybe looking into a reverse proxy like Caddy or NGINX to plug in front of the service itself to have this functionality in place.

BEE-22 Lack of Secure Random Number Generation

● High ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `7448289d52a75edf2a66601224f876b6b3882e2f`.
The client provided the following explanation:

> Used randomBytes instead of Math.random and used whole capital alphabet to extend the set (lower the duplicate key odds)

Description: The `generateUserCode` function in the `deviceAuth.service.ts` file is not using a secure random number generator from the crypto library. Instead, it appears to be generating random numbers using a less secure method, which could make the generated codes predictable and vulnerable to brute-force attacks.

Recommendation: Replace the current random number generation mechanism with a secure alternative from the Node.js crypto library.

BEE-23 Insufficient per-user limitation of LLM queries

● High ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.
The client provided the following explanation:

> Rely on load balancer for rate limiting

Description: The system currently lacks adequate rate limiting mechanisms for Large Language Model (LLM) queries initiated by users. Security testing revealed that users can execute multiple parallel requests towards the stream endpoints without encountering any restrictions or throttling. This could potentially lead to resource exhaustion, wallet-draining or denial of service attacks.

Recommendation: 1. Implement a per-user rate limiting mechanism to control the frequency and concurrency of LLM queries.
2. Establish reasonable limits based on typical usage patterns and resource constraints.
3. Store information about currently open streams or active queries for each user, possibly in a data store like Redis.
4. Block or queue new requests from a user if they exceed the defined rate limits until their existing queries are completed.
5. Monitor and adjust rate limits periodically based on system performance and user feedback.

BEE-24 Insecure OAuth Implementation

● High ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `e96ce4ec91e9bcb51cd9d21c1c04b5b1b6298372` .
The client provided the following explanation:

> dropped unused insecure endpoints for MCP OAuth

Description: The OAuth implementation in the `beep/beep-server-dev/src/mcp/oauth.ts` file has several security vulnerabilities. The endpoints responsible for authentication and authorization do not perform proper checks, leaving the system vulnerable to unauthorized access and potential account hijacking. Additionally, there is a risk of client ID collision during the registration process.

Recommendation: 1. Implement proper authentication checks for all OAuth endpoints to ensure that only authorized clients and users can access sensitive operations.
2. Enhance the client ID generation and verification process during registration to prevent collisions and potential impersonation attacks.
3. Review and strengthen the authorization logic in the `/authorize` endpoint to ensure that access is granted only after proper validation and user consent.
4. Implement best practices for OAuth security, such as using secure communication channels, storing sensitive data securely, and implementing proper access control mechanisms.
5. Conduct thorough security testing and code reviews to identify and mitigate any other potential vulnerabilities in the OAuth implementation.

BEE-25 Match local Redis security settings to Production

● Medium ⓘ Fixed

Alert

Marked as "Unresolved" by the client.
The client provided the following explanation:

> We have updated to the use of redis url. Is this still required?

Description: We recommend to have local testing environments to match production as much as possible, in order to detect any potential infrastructure related issues early on. Hence, setting up Redis with all mandatory security features is recommended.

Recommendation: For your Redis Docker image, we recommend the following changes.

- In the `redis.conf` file, set the parameter `aclfile` to a path to a file containing the ACL instructions.
- In the `redis.conf` file, set the parameter `requirepass` , if you cannot set `aclfile` .
- Set the parameter `tls-replication` to `yes` (default is `no`).
- In the `redis.conf` , set the parameter `tls-port` to any other value but `0` , and ensure that the parameters `tls-cert-file` and `tls-key-file` are set.
- Inside the `redis.conf` , set the parameter `tls-protocols` to `TLSv1.2 TLSv1.3` (unclear default).
- In the `redis.conf` , set the parameter `enable-protected-configs` to `yes` .
- In the `redis.conf` , set the parameter `bind` to specific IP addresses, and avoid listening to traffic from wildcard IP addresses (default).

- Inside the `redis.conf`, set the parameter `tls-prefer-server-ciphers` to `yes` (default is `no`).
- In the `redis.conf`, Set the parameter `timeout` to any other value but `0`.

BEE-26 Match local Postgres security settings to Production

● Medium ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `02c9a17a470b55329f21fb73697521292b3920bf`.
The client provided the following explanation:

> Added ssl config

Description: We recommend to have local testing environments to match production as much as possible, in order to detect any potential infrastructure related issues early on. Hence, setting up Postgres with all mandatory security features is recommended.

Recommendation: For the Postgres Docker image, we recommend the following changes.
- Set the `ssl` directive to `on` and set the respective certificate files.

BEE-27 Pin actions to SHA sum to prevent supply chain attacks

● Medium ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `c39d5837c496a04fb2895b22ea6b8d6d499aafab`, `5410d815058da0186e83c7b55c3a2d418c94e4c4`, `0fccda6dde18afb1890b1185c8255c18e82abd0a`.
The client provided the following explanation:

> pinned sha

Description: When using external actions in your workflow, it is important to ensure that the version remains the same to avoid run-failures.

Recommendation: For every `uses` instruction in your GitHub actions, ensure that it is referenced by its SHA hash, and not just the version tag.

BEE-28 Logical code inconsistencies

● Medium ⓘ Fixed

Update
Marked as "Fixed" by the client.
Addressed in: `19adce3b6e5436922c7320046da6bca6755bddb8`, `8a36601a4545398c643103648721ec885e62cd2f`.
The client provided the following explanation:

> update code in various places to address logical inconsistencies

Description: A review of the codebase revealed multiple logical inconsistencies that may impact reliability, maintainability, and correctness. These issues include:

- Variables treated as potentially null or undefined in one area, but subsequently dereferenced without checks in the same or adjacent code paths.
- Redundant or identical evaluation branches that indicate copy-paste errors or misunderstood control flow.
- Dead or unreachable code paths resulting from conditions being evaluated earlier in the function, making later checks or logic blocks ineffective.

Such inconsistencies increase the likelihood of runtime exceptions, unintended behavior, and subtle bugs that may be difficult to detect through casual testing. They also add cognitive overhead for maintainers and may hide deeper architectural or data-flow issues.
- `beep-server-dev/src/grpc/services/mappers/transaction-info-extractor.ts`, line 48: accessing `object.object.object_type`, implicitly assuming `object.object` is not `null`-like object, and then in line 59 using the questionmark operator, implying that it could be `null`.
- `beep-server-dev/src/services/sui.service.ts`, line 190: `sponsorWalletNeedsRebalancing` assumes a non-null input value, but line 194 indicates that you are expecting potentially a `null` for `sponsorCoins`.
- `beep-server-dev/src/webhooks/privy/privy-controller.ts`, line 235: mapping over `referencedInvoiceLineItems`, but in line 236 there is the questionmark operator, indicating that the value may be `null`.
- `beep-server-dev/src/utils/db-models.util.ts`, line 108: `model` may be `null` as set in line 107 where the questionmark operator is used.

- `beep-server-dev/src/services/payment.service.ts`, line 401: Accessing length of `uniqueProductsByMerchant`, while in line 419 using the question mark operator, indicating that it may be `null`.
- `beep-frontend/src/components/common/CommonDropdown.tsx`, line 134: The same conditional output is chosen independent of the value of `isOpen`.
- `beep-server-dev/src/controllers/product.controller.ts`, line 57: There needs not to be an alternative assignment for `merchantId`, since `authMerchantId` is already verified to be non-null as of line 33.
- `beep-frontend/src/components/funds/steps/StepWallet.tsx`, line 47-48: `sortedWallets` is assumed non-null, but in line 54 the questionmark operator indicates that it may be `null`.
- `beep-server-dev/src/grpc/services/sui-checkpoint-processing-service.ts`, line 48: In the same line, `trx` may be assumed to be `null` or not.
- `beep-server-dev/src/services/payment.service.ts`, line 917: `payoutResult` directly accessed, but in line 923 a questionmark operator is used.
- `beep-server-dev/src/__tests__/solana.service.test.ts`, lines 83-95: Dead test, since a `return` is stated at the beginning of the test.
- `beep-frontend/src/components/user-account-dropdown/AccountSettings/NewAPIKeyAlert.tsx`, line 44: `copySuccess` is already considered `false` in this branch (see line 39).
- `beep-frontend/src/components/user-account-dropdown/DefaultDropdown.tsx`, line 237: `isMobile` is `false` in this branch.
- `beep-server-dev/src/services/treasury/treasury-withdrawal.service.ts`, line 482: `withdrawRequest` could be null on this line.
- `beep-server-dev/src/services/treasury/treasury-reallocation.service.ts`, line 200: `currentData` not checked for null, but checked in line 196.
- `beep-server-dev/src/webhooks/alchemy-pay/alchemy-pay.controller.ts`, line 139: `referencedInvoiceLineItems` not checked for null, but checked for null in line 154.
- `beep-server-dev/src/services/code-execution/code-execution.service.ts`, line 209: Derlerencing while using a question mark operator in line 198.
- `beep-server-dev/src/grpc/services/mappers/sui-mapper.ts`, line 60: The `inputTrx?` should be on the left side?
- `beep-server-dev/src/tools/charts/generate-chart.tool.ts`, line 170: Accessing `executionResult` properties, even though the line 151 indicates that it could be `null`.
- `beep-server-dev/src/controllers/user.controller.ts`, line 25: `merchantId` as at this line already guaranteed to be non-null.
- `beep-server-dev/src/grpc/services/sui-indexer.service.ts`, line 182: `trx` is treated partially as if it can be null, and partially if it can't be.
- `beep-server-dev/src/services/treasury/treasury-allocation.service.ts`, line 275: `treasuryAccount` is assumed to be non-null, but in line 278, `null` is marked as possibility.
- `beep-server-dev/src/services/payment.service.ts`, line 669: `invoiceLineItems` not assumed to be `null`, but in line 670, marked as possible to be `null`.
- `beep-server-dev/src/services/bridge/relay.provider.ts`, line 112: `executeResponse` dereferenced, but question mark operator used in lines 122 and following.
- `beep-server-dev/src/services/treasury/allocation-context-service.ts`, line 237: Treating `snapshots` like a non-null value, but treated earlier as possible to be `null` in line 189
- `beep-server-dev/src/utils/ai-models.util.ts`, line 126: Dereferencing `model`, but in line 138 marked as possible to be `null`.
- `beep-server-dev/src/grpc/services/mappers/sui-mapper.ts`, line 181: `split_coins.coins.coin` dereferenced, but marked as possible null in line 184.
- Why is `active` equals to `true` if `merchant.active` is `undefined`? (`beep-server-dev/src/services/merchant.service.ts`, line 80)
- Mix of boolean-able types: `beep-sdk-dev/packages/checkout-widget/src/QueryProvider.tsx`, line 8. One option assigns `1`, the other a boolean.

Recommendation:

- Perform a systematic review of affected functions to verify variable assumptions, ensuring consistent nullability handling and proper type narrowing.
- Refactor redundant or duplicate conditional branches to simplify control flow and remove ambiguity.
- Remove dead or unreachable code while verifying that the intended logic is preserved.
- Strengthen TypeScript typing (e.g., enabling strictNullChecks, improving discriminated unions, refining interfaces) to allow the compiler to catch such inconsistencies.
- Introduce static analysis tools to automatically detect logical inconsistencies and unreachable code in future commits.
- Add or enhance unit tests for critical functions to validate expected execution paths and guard against regressions.

BEE-29 Presence of Unresolved TODO Comments in Codebase

● Medium ⓘ Mitigated

Update

TODO items have been transformed into tickets for management visibility and analytics.

Alert

Marked as "Unresolved" by the client.

The client provided the following explanation:

> Not a top priority regarding todos

Description: During the review of the source code repository, several TODO comments were identified across multiple files. These comments typically indicate incomplete functionality, deferred decisions, or known issues that developers intended to revisit. Leaving TODOs in production-relevant branches introduces ambiguity regarding the implementation's completeness, may obscure latent defects, and may hinder maintenance efforts. In some cases, TODOs can also signal missing validations, hardening steps, or other security-relevant work that has not yet been completed. As such, the presence of unaddressed TODOs may reduce code reliability and overall engineering assurance.

Some examples are:

- `beep-frontend/beep-frontend-654879b7050b15240a2aadfdfd55a8ebd7392369/src/hooks/useExport.ts`
- `beep-sdk-dev/packages/cli/templates/server/src/mcp-server.ts`
- `beep-server-dev/src/__tests__/product.test.ts`
- `beep-server-dev/src/__tests__/invoice.test.ts`
- `beep-server-dev/src/__tests__/request.test.ts`
- `beep-server-dev/src/controllers/invoice.controller.ts`
- `beep-server-dev/src/controllers/payment.controller.ts`
- `beep-server-dev/src/controllers/toSController.ts`
- `beep-server-dev/src/controllers/wallet-balance.controller.ts`
- `beep-server-dev/src/db/migrations/000001_2025-08-21_124200__initial-script.js`
- `beep-server-dev/src/db/models/InvoiceLineItem.ts`
- `beep-server-dev/src/db/models/TreasuryAccount.ts`
- `beep-server-dev/src/db/models/Price.ts`
- `beep-server-dev/src/db/repositories/transaction.repository.ts`
- `beep-server-dev/src/grpc/services/mappers/sui-mapper.ts`
- `beep-server-dev/src/grpc/services/mappers/transaction-info-extractor.ts`
- `beep-server-dev/src/grpc/services/sui-consolidation-service.ts`
- `beep-server-dev/src/services/buyer-email-verification/buyer-email-verification.service.ts`
- `beep-server-dev/src/services/cron.service.ts`
- `beep-server-dev/src/services/helius/helius.service.ts`
- `beep-server-dev/src/services/onramp/onramp.service.ts`
- `beep-server-dev/src/services/payout.service.ts`
- `beep-server-dev/src/services/payment.service.ts`
- `beep-server-dev/src/services/privy.service.ts`
- `beep-server-dev/src/services/reconciliation/reconciliation.service.ts`
- `beep-server-dev/src/services/treasury/common.ts`
- `beep-server-dev/src/services/treasury/treasury-reallocation.service.ts`
- `beep-server-dev/src/services/treasury/treasury-allocation.service.ts`
- `beep-server-dev/src/services/treasury/treasury.service.ts`
- `beep-server-dev/src/services/treasury/yield-generator-providers/kamino-vault-yield-generator.ts`
- `beep-server-dev/src/services/solana.service.ts`
- `beep-server-dev/src/services/treasury/treasury-withdrawal.service.ts`
- `beep-server-dev/src/services/turnkey.service.ts`
- `beep-server-dev/src/services/wallet-provider.service.ts`
- `beep-server-dev/src/services/wallet.service.ts`
- `beep-server-dev/src/tools/types.ts`
- `beep-server-dev/src/tools/wallet/sign-solana-transaction.tool.ts`
- `beep-server-dev/src/tools/wallet/request-and-purchase-asset.tool.ts`
- `beep-server-dev/src/tools/wallet/streaming/check-payment-status.tool.ts`
- `beep-server-dev/src/utils/chains.ts`
- `beep-server-dev/src/utils/db-models.util.ts`
- `beep-server-dev/src/utils/tokens.ts`
- `beep-server-dev/src/webhooks/alchemy-pay/alchemy-pay.controller.ts`
- `beep-server-dev/src/webhooks/privy/privy-controller.ts`

Recommendation: Review all TODO comments across the repository and classify them by type (e.g., missing feature, technical debt, security concern, refactor).

For each TODO, either:

- Implement the missing functionality or corrective action, or
- Convert it into a tracked work item in the project's issue tracker and remove the inline TODO from the source.
- Integrate a repository rule or CI check (e.g., linting or commit hooks) to prevent new TODOs from being merged without a corresponding issue reference.
- Establish engineering guidelines defining how deferred work should be documented and tracked to avoid accumulation of ungoverned technical debt.

BEE-30

Risk of Reverse Tabnapping Due to Use of target="_blank" Without rel Attributes

● Medium ⓘ Fixed

Update

Marked as "Fixed" by the client.

Addressed in: `8e5994f40a8c9f8f9667800df5a48b7a8c22ec9c`.

The client provided the following explanation:

> prevent reverse tab napping

Description: A review of the front-end code revealed multiple instances where hyperlinks open in a new browser tab using target="_blank" but do not specify a corresponding rel="noopener" or rel="noreferrer" attribute. Without these attributes, the newly opened page receives a window.opener reference to the originating application. A malicious or compromised external page could exploit this by manipulating the window.opener object to redirect the original tab to a phishing site, malware distribution page, or malicious login prompt. This class of attack—known as reverse tabnapping—poses a reputational and user-safety risk and can undermine trust in the application interface.

Affected Files:

- `/beep-server-dev/src/tools/auth/signup.html`, set `rel` to `noopener`.
- `/beep-frontend/src/components/RewardsPromotion.tsx`, line 100

Recommendation: - Add rel="noopener" or rel="noopener noreferrer" to all hyperlinks that use target="_blank".

- Update UI component libraries or shared link components to automatically include secure rel attributes when opening new tabs.
- Perform a scan or linting check to ensure no new instances of unsafe target="_blank" usage are introduced into the codebase.
- Document this as a front-end security requirement in the engineering standards to ensure ongoing compliance.

BEE-31 Drift from use of ORM

● Medium ⓘ Acknowledged

Update

Marked as "Acknowledged" by the client.

The client provided the following explanation:

> This is direct SQL is necessary for a function within our application and we believe not a direct concern for SQL injection

Description: In the `beep-server-dev/src/services/treasury/treasury-recreation.service.ts` file, a query with string formatting is used instead of the Object-Relational Mapping (ORM) functionality employed throughout the codebase. This practice introduces the risk of SQL injection vulnerabilities, which can potentially allow an attacker to execute arbitrary SQL commands on the database.

```
  private getCurrentDbState(treasuryAccountId: number) {
    return sequelize
      .query<{ yield_generator: string; total_shares: string }>(
        `
          SELECT
            tb.yield_generator,
            COALESCE(SUM(tb.deposit_shares), 0) as total_shares,
            COALESCE(SUM(tb.current_amount), 0) as total_amount,
            COALESCE(SUM(tb.current_amount_decimal), 0) as total_amount_decimal,
            COUNT(tb.id) as block_count
          FROM treasury_blocks tb
          WHERE tb.treasury_account_id = ${treasuryAccountId}
            AND tb.status = 'ALLOCATED'
            AND tb.deleted_at IS NULL
          GROUP BY tb.yield_generator
     `,
        { type: QueryTypes.SELECT, raw: true },
      )
      .then((r) => (Array.isArray(r) ? r : [r]))
      .then((r) => r.reduce((acc, c) => ({ ...acc, [c.yield_generator]: c }), {}));
```

Recommendation: 1. Refactor the affected code to utilize the ORM functionality consistently across the application.

2. Ensure that user input or external data is properly sanitized and validated before being used in database queries.
3. Implement input validation mechanisms to prevent malicious data from being passed into the query.
4. Consider using parameterized queries or prepared statements provided by the ORM to mitigate SQL injection risks.
5. Conduct thorough code reviews and security testing to identify and address any remaining instances of insecure query usage.

BEE-32 Disable Default GCP Network

● Medium ⓘ Fixed

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

> resolved in GCP

Description: When creating a project in the Google Cloud Platform, there is an option to auto-generate a default network to be used for applicable resources which do not attach to a specified network. This default network allows automatically ICMP, RDP and SSH traffic into the network. Furthermore, logging and other custom required compliance configurations are disabled for this default network.

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

BEE-33 Enable DNSSEC with Modern Signing Key for GCP DNS Managed Zone

● Medium ⓘ Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

> DNSSEC can only be enabled on public zones.
> So, changes are applied only to beepitprod-public (and also in dev)

Description: It is generally recommended to enable DNSSEC for any DNS managed zone in Google Cloud, and ensure that the signing key does not use `rsasha1`.

Affected Resources:
- `beepitprod-private`
- `beepitprod-public`

Recommendation: For every resource of type `google_dns_managed_zone`, ensure that the `dnssec` config block exists, and that the `state` is set to `on`. Furthermore, inside the `dnssec_config` block, ensure that inside the `default_key_specs` block does not have `rsasha1` defined as `algorithm`.

BEE-34 Restrict GCP Firewall Source Ranges

● Medium ⓘ Acknowledged

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

> Removed the default-*.
> The source 0.0.0.0/0 is used only in VPN firewall rule.
> This is needed, because it's a VPN relay and the client source IP is unknown.
> In this case, it's an accepted risk

Description: If the source is defined using `source_ranges` for a Google firewall rule, one needs to ensure that the range is never set to the CIDR block `0.0.0.0/0`.

Affected Resources:
- `default-allow-icmp`
- `default-allow-rdp`
- `default-allow-ssh`
- `production-allow-internal`
- `production-allow-vpn`

Recommendation: Remediation: For every resource of type `google_compute_instance`, ensure that the array defined by `source_ranges` does not contain `0.0.0.0/0`.

BEE-35 Ensure GCP Project Audit Logging Enabled

● Medium ⓘ Fixed

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

> This was added in our new repo infra-terraform
> commit: 831a3afa95fcea3c9b3cdf06f44d22c8b6094845

Description: It is recommended that audit logging is happening on all services, tracking read and write activities.

Recommendation: For every project, ensure that there is a corresponding `google_project_iam_audit_config`, with the key `service` set to `allServices`, and three `audit_log_config` blocks, with the respective log types

- `ADMIN_READ`,
- `DATA_READ`, and
- `DATA_WRITE`.

Additionally, in each of the `audit_log_config` blocks, there should be no `exempted_members` list present.

BEE-36 Enable GCP Container Image Vulnerability Scanning

• Medium ⓘ Fixed

Update
Marked as "Acknowledged" by the client.
The client provided the following explanation:

> This was added in our new repo infra-terraform
> 535238c94e104956fc76e0101f99a9da179b5b2a

Description: Google has a built-in service to scan container images for vulnerabilities. We generally recommend to use it.

Recommendation: Ensure that there is at least one Google cloud resource of type `google_project_service`, where the `service` key is set to `containerscanning.googleapis.com`.

BEE-37 Input Length Restrictions Lacking

• Medium ⓘ Fixed

Update
While we observed several code instances where input lengths are being validated and checked against, it is recommended to define the restriction on the Swagger documentation layer, where it would hence be blocked off as early in the process as possible.

Update
Marked as "Fixed" by the client.
Addressed in: `652d32bdbadf779fca36498a5ef0284f9ab253b6`.
The client provided the following explanation:

> Set various max input lengths and leverage middleware for validation

Description: The application's APIs do not consistently enforce length restrictions on input data. Allowing unbounded input lengths can lead to denial of service (DoS) attacks or buffer overflows, which can compromise system availability and potentially expose sensitive data.

Recommendation:

- Implement input validation mechanisms to enforce appropriate length limits on all user-supplied data across all APIs.
- Define and document maximum allowed lengths for each input field based on the expected use case and data type.
- Reject or truncate inputs that exceed the defined length limits, and return appropriate error messages to the client.
- Regularly review and update input length restrictions as application requirements evolve.

BEE-38 Lack of Pagination in Queries

• Medium ⓘ Fixed

Update
Resolved in later commit during re-audit.

Alert

Marked as "Unresolved" by the client.
The client provided the following explanation:

> We believe this doesn't pose an exponential risk at this point. Specifically because the queries that could be impacted are already restricted to s specific user or wallet

Description: Queries without proper pagination or limits can potentially retrieve an excessive amount of data, leading to performance issues, resource exhaustion, and potential security vulnerabilities. Even when a date range is specified, large datasets can still cause problems if pagination is not implemented correctly.

Recommendation:

- Implement pagination mechanisms for all queries, regardless of whether a date range is specified or not.
- Set reasonable limits on the number of records returned per query, such as 100 or 1000 records, depending on your application's requirements.
- Provide pagination controls in the user interface, allowing users to navigate through the result set efficiently.
- Consider implementing server-side checks to enforce maximum record limits, preventing clients from requesting an excessive amount of data.

BEE-39 Sensitive Implementation Details Exposure

● Medium ⓘ Fixed

Update

Marked as "Fixed" by the client.
Addressed in: `5942f33fbb69334d2e1348f00ac6e6c1fbf10bee`.
The client provided the following explanation:

> added centralized error handling

Description: The application's API endpoints expose sensitive implementation details when returning error responses. These implementation details, such as internal server paths, code snippets, or stack traces, can provide valuable information to attackers, assisting them in identifying vulnerabilities or weaknesses in the application's design and architecture.

Example command and output:

```
curl https://api.dev.justbeep.it/v1/money-talks-agent/stream\?
query\=how%20much%20is%20my%20balance%3F\&messages\=%5B%5D

{"success":false,"error":{"code":"AppError","message":"Authentication required","details":
{"type":"AppError","stack":"AuthenticationError: Authentication required\n    at authenticate
(/usr/src/app/dist/src/middlewares/auth.middleware.js:12:11)\n    at
/usr/src/app/node_modules/@opentelemetry/context-async-
hooks/build/src/AbstractAsyncHooksContextManager.js:46:55\n    at AsyncLocalStorage.run
(node:internal/async_local_storage/async_hooks:91:14)\n    at AsyncLocalStorageContextManager.with
(/usr/src/app/node_modules/@opentelemetry/context-async-
hooks/build/src/AsyncLocalStorageContextManager.js:33:40)\n    at contextWrapper
(/usr/src/app/node_modules/@opentelemetry/context-async-
hooks/build/src/AbstractAsyncHooksContextManager.js:46:32)\n    at patched
(/usr/src/app/node_modules/@opentelemetry/instrumentation-express/build/src/instrumentation.js:224:73)\n
at Layer.handle [as handle_request] (/usr/src/app/node_modules/express/lib/router/layer.js:95:5)\n    at
trim_prefix (/usr/src/app/node_modules/express/lib/router/index.js:328:13)\n    at
/usr/src/app/node_modules/express/lib/router/index.js:286:9\n    at Function.process_params
(/usr/src/app/node_modules/express/lib/router/index.js:346:12)","isOperational":true,"requestId":"6ee9a0b
b-6de2-4833-84ef-5557479b324f"}}}
```

Recommendation:

- Implement a centralized error handling mechanism that sanitizes and filters out sensitive information from error responses.
- Return generic, user-friendly error messages that do not reveal internal implementation details.
- Enable error logging for detailed debugging purposes, but ensure that these logs are properly secured and not accessible from the public-facing application.
- Regularly review and update error handling practices to align with industry best practices and emerging threats.

BEE-40 Persist operating system logs in Docker images

● Low ⓘ Unresolved

Alert
Marked as "Unresolved" by the client.
The client provided the following explanation:

> Determined that low priority issues can be resolved at a later date

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 which has `/var/log` as an argument.

BEE-41 Enable healthchecks for all Docker images

• Low ⓘ Unresolved

Alert
Marked as "Unresolved" by the client.
The client provided the following explanation:

> Determined that low priority issues can be resolved at a later date

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.

BEE-42 Unsafe Temporary File Creation Without Using a Proper Tempfile Library

• Low ⓘ Unresolved

Alert
Marked as "Unresolved" by the client.
The client provided the following explanation:

> Determined that low priority issues can be resolved at a later date

Description: The codebase includes logic that manually generates temporary file names and writes to those paths without using a dedicated, collision-resistant tempfile library. This approach relies on assumptions about file uniqueness and system timing, which may not hold under concurrent execution, high-load scenarios, or adversarial conditions. Manual tempfile creation introduces risks such as filename collisions, race conditions, or unintended overwriting of existing files. In scenarios where the temp directory is accessible to other processes or users, it may also enable symlink attacks or unauthorized file manipulation. Modern TypeScript/Node.js best practices recommend using well-vetted libraries that handle secure, atomic creation of temporary files.

Affected Files:
- `beep-server-dev/src/controllers/transaction.controller.ts`, line 664: File may already exist, you may overwrite it, etc.

Recommendation:
- Replace all manual tempfile creation logic with a secure tempfile library (e.g., tmp, tmp-promise, tempy, or Node's built-in fs.mkdtemp for directories).
- Ensure tempfiles are created atomically and with appropriate permissions to prevent race conditions or unauthorized access.
- Implement automatic cleanup mechanisms provided by these libraries to avoid orphaned tempfiles.
- Add coding standards or lint rules requiring use of approved tempfile utilities for any future temporary file operations.

BEE-43 Enable Compute Subnetwork Flow Logs

• Low ⓘ Unresolved

Alert
Marked as "Unresolved" by the client.
The client provided the following explanation:

> Determined that low priority issues can be resolved at a later date

Description: For any compute subnetwork, the flow logs are a great tool for debugging and attack detection.

Recommendation: For every resource of type `google_compute_subnetwork`, ensure that the `log_config` block exists.

BEE-44 Restrict Public IP Access to GCP Compute Instances

● Low ⓘ Unresolved

Alert
Marked as "Unresolved" by the client.
The client provided the following explanation:

> Determined that low priority issues can be resolved at a later date

Description: Google compute instances should be part of a cluster, and hence any public endpoint should be associated with a load balancer of some kind. Allowing access to a compute instance directly via the internet is not recommended.

Affected Files:
- `vpn-prod`

Recommendation: For every resource of type `google_compute_instance`, ensure that in the `network_interface` block, the keys `access_config` and `ipv6_access_config` do not exist.

BEE-45 Header Leaks Implementation Details

● Low ⓘ Unresolved

Alert
Marked as "Unresolved" by the client.
The client provided the following explanation:

> Determined that low priority issues can be resolved at a later date

Description: The `X-Powered-By` HTTP response header exposes the technology stack used by the server, revealing implementation details that may aid potential attackers in identifying and exploiting known vulnerabilities. This header is currently being sent by the server, as evident in the `/beep-server-dev/src/server.ts` file, line 48.

Recommendation: Disable the `X-Powered-By` header by removing or commenting out the line of code that sets this header. This can typically be done by either removing the corresponding middleware or setting the header value to an empty string.

BEE-46 Inefficient Multi-Layer Docker Build

● Low ⓘ Unresolved

Alert
Marked as "Unresolved" by the client.
The client provided the following explanation:

> Determined that low priority issues can be resolved at a later date

Description: The Dockerfile for the `beep-frontend` application employs a multi-layer build strategy. However, this approach may not be optimal in this case as both the build and dependency layers use the same base image, and the primary large file is shared between them, resulting in redundant layers and increased image size.

Recommendation: Consolidate the build process into a single layer by combining the steps for installing dependencies and copying the application code.

BEE-47 Duplicated Conditional Logic in ChatContext

● Low ⓘ Unresolved

> **Alert**
> Marked as "Unresolved" by the client.
> The client provided the following explanation:
>
> > Determined that low priority issues can be resolved at a later date

Description: The `ChatContext.tsx` file within the `beep-frontend/src/app/context` directory contains duplicated conditional logic in lines 49-51. Repeating the same conditions across multiple locations in the codebase can lead to maintainability issues, as any future changes to the conditions will require modifications in multiple places.

Recommendation: Refactor the duplicated conditional logic into a separate function or utility method. This will improve code readability, maintainability, and reduce the likelihood of introducing bugs when modifying the conditions in the future. Follow these steps:

1. Identify the repeated condition(s) in lines 49-51 of `ChatContext.tsx`.
2. Create a new function or utility method that encapsulates the conditional logic.
3. Replace the duplicated conditions in `ChatContext.tsx` with a call to the newly created function or utility method, passing any necessary arguments.
4. Test thoroughly to ensure the refactored code behaves as expected.

BEE-48 Centralize Type Definitions

• Low ⓘ Unresolved

> **Alert**
> Marked as "Unresolved" by the client.
> The client provided the following explanation:
>
> > Determined that low priority issues can be resolved at a later date

Description: Type definitions are scattered throughout the codebase, leading to duplication and potential inconsistencies. For instance, the `activeToolTip` type is defined in both `beep-frontend/src/components/Header/TotalYieldPill.tsx` and `beep-frontend/src/components/Header/LiveYieldPill.tsx`. This approach can make it difficult to maintain and update types consistently across the application.

Recommendation: 1. Identify all common types used across multiple components or modules.
2. Create a dedicated directory or file (e.g., `types.ts`) to centralize the definitions of these common types.
3. Import and use the centralized type definitions wherever needed throughout the codebase.
4. Update the existing code to use the centralized type definitions, removing any duplicated or scattered type definitions.
5. Establish a process or guidelines to ensure that new types are added to the centralized location and used consistently across the codebase.

BEE-49 Use of Integer Type for Unique Identifiers

• Low ⓘ Unresolved

> **Alert**
> Marked as "Unresolved" by the client.
> The client provided the following explanation:
>
> > Determined that low priority issues can be resolved at a later date

Description: The codebase utilizes integer types for unique identifiers, which may lead to potential issues in the long run. While integers can provide acceptable performance for identifiers, their limited range can pose risks as the database grows larger. Additionally, the use of integers for identifiers is generally discouraged due to the potential for collisions and the lack of inherent meaning in the values.

Affected Files:

- `beep-server-dev/src/db/models/BuyerEmailVerification.ts`, line 68

Recommendation: Replace instances where integers are used for unique identifiers with the `bigint` type. The `bigint` type provides a larger range and is better suited for use as unique identifiers, reducing the risk of collisions and providing better scalability as the database grows. Additionally, consider adopting a more meaningful and structured approach to generating unique identifiers, such as UUIDs or sequential GUIDs, which can enhance maintainability and provide additional context.

BEE-50 Inefficient Delay in Treasury Withdrawal Service

• Low ⓘ Unresolved

> **Alert**

Marked as "Unresolved" by the client.
The client provided the following explanation:

> Determined that low priority issues can be resolved at a later date

Description: The Treasury Withdrawal Service in the `beep-server-dev/src/services/treasury/treasury-withdrawal.service.ts` file uses a `sleep(1000)` function to introduce a delay of 1 second. This approach is fragile and inefficient, as the required delay time may vary depending on the system load and other factors, potentially causing issues or delays in the withdrawal process.

Recommendation: Instead of using a fixed delay with `sleep(1000)`, implement an asynchronous approach using `await` or Promises. This will allow the service to wait for the necessary conditions to be met before proceeding with the withdrawal process, without introducing unnecessary delays or blocking the event loop.

BEE-51
Chat history stored and communicated through URL query parameters

• Low ⓘ Unresolved

Alert
Marked as "Unresolved" by the client.
The client provided the following explanation:

> Determined that low priority issues can be resolved at a later date

Description: The `money-talks-agent/stream` endpoint propagates session history through URL parameters, which can pose a security risk. Sensitive data transmitted via URL parameters may be exposed through server logs, browser history, or other unintended channels. Additionally, URLs have a size limit, which could potentially lead to truncation or loss of crucial data.

Recommendation:
- Refactor the application to transmit session history as part of the request payload instead of URL parameters.
- Enforce strict input validation and sanitization on both client and server sides to prevent potential injection attacks.
- Consider implementing session management mechanisms that do not rely on transmitting sensitive data through URLs or query strings.

BEE-52 Define a timeout for any GitHub Actions

• Informational ⓘ Unresolved

Alert
Marked as "Unresolved" by the client.
The client provided the following explanation:

> Determined that low priority issues can be resolved at a later date

Description: Jobs in GitHub actions should run and potentially fail quickly. Setting an expected time-limit can be useful to ensure performance goals.

Recommendation: For every job inside the `job` section of a workflow, ensure that the key `timeout-minutes` is set.

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).

Changelog

- 2025-11-25 - Initial 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 1100 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, Solana, Polygon, TON, Cardano, Binance Smart Chain, Avalanche, Arbitrum, Flow
- DeFi: Lido, Ethena, Compound, Curve, Venus, PancakeSwap, Polymarket
- Infra/Staking: TrustWallet, Alchemy, Liquid Collective, Kiln, Galxe, API3, ssv.network, Luganodes
- Immersive: Virtuals Protocol, Wayfinder, OpenSea, Square Enix, Parallel, Camp, Decentraland
- Institutional: Visa, Circle, Revolut, Anthropic, OpenAI, Canton, Republicc, Arkham, Sequoia

any decisions to buy or sell a product, service or any other asset. Quantstamp does not warrant, endorse, guarantee, or assume responsibility for any product or service advertised or offered by a third party, or any open source or third-party software, code, libraries, materials, or information to, called by, referenced by or accessible through the report, its content, or any related services and products, any hyperlinked websites, or any other websites or mobile applications, and we will not be a party to or in any way be responsible for monitoring any transaction between you and any third party. As with the purchase or use of a product or service through any medium or in any environment, you should use your best judgment and exercise caution where appropriate.

Quantstamp

Beep