

Full Security Report

1. Executive Summary

This report contains details pertaining to the security posture of ReliefConnect, specifically addressing application-layer, network-layer, and system-layer security. A comprehensive security assessment, including penetration testing and vulnerability analysis, was conducted to evaluate the platform's defenses. An assessment was performed on April 24th, 2025, at 10:00 - 19:00 CDT. The scope of the test included ReliefConnect's web application and APIs, authentication mechanisms (Google OAuth, CAPTCHA), data storage and transmission (MongoDB, HTTP/HTTPS), and CI/CD pipeline security.

ReliefConnect has identified and categorized potential threats based on the OWASP Top 10 and OWASP API Security Top 10:

- Injection (e.g., XSS, SQLi)
- Broken Authentication (OAuth hijacking attempts)
- Sensitive Data Exposure (MITM attacks via HTTP)
- Denial of Service (DoS) (API flooding)
- Misinformation/Abuse (fake Safe & Well entries)

We found 7 vulnerabilities during our assessment of ReliefConnect's assets: **1 critical**, **2 high**, **2 moderate**, **1 low**, and **2 Secure/Informational**. To maintain data confidentiality, integrity, and availability, ReliefConnect should work on fixing the vulnerabilities presented in our security assessment findings.

Leaving these systems in their current state will expose them to the risk of an intrusion, which would lead to severe fines, legal consequences, and loss of consumer trust. It is highly recommended that ReliefConnect reviews the detailed list of vulnerabilities located further on in this document and begins remediation immediately.

2. Use and Abuse Cases

Use Cases

| Actor | Goal | Flow |
|---------------------------------------|-------------------------------------|---|
| Disaster-affected individual | Register as Safe and Well | Navigate → Enter name, phone, address, message → Submit form → Receive confirmation. |
| Concerned family member | Search for a loved one | Access search → Enter last name + phone/address → View status and message. |
| Disaster-affected individual | Locate nearby resources | Access resource map → Apply filters (shelter, aid) → View real-time API data. |
| Authenticated user | Participate in community discussion | Log in via Google OAuth → Post/respond to local updates. |
| Any user (accessing restricted areas) | Secure login via Google OAuth | Select login → Complete OAuth flow → Access restricted features (e.g., community chat). |

Abuse Cases

| Actor | Goal | Flow |
|-------------------------|---------------------------------|--|
| Malicious bot | Bypass CAPTCHA for spam | Bypass CAPTCHA client-side → Submit bulk fake registrations/posts. |
| Network attacker (MITM) | Intercept unencrypted HTTP data | Monitor traffic → Capture PII transmitted over HTTP. |
| Skilled attacker | Unauthorized API access | Manipulate API endpoints → Read/modify unauthorized data. |

| | | |
|-----------------------|-------------------------------|---|
| Malicious user | Inject XSS in community posts | Submit post with JavaScript → Scripts execute when viewed by others. |
| Attacker/botnet | DoS via API flooding | Flood endpoints (e.g., search, maps) → Exhaust resources or third-party API limits. |
| Malicious user | Spread misinformation | Register fake Safe & Well entries → Mislead others with false safety statuses. |
| MITM/malicious script | OAuth token hijacking | Intercept session tokens (if insecure) → Reuse to impersonate user. |

3. Recommended Immediate Changes

Listed below are observations we made while conducting the vulnerability assessment. These are intended to be “recommended improvements” and follow industry best practices.

- Enforce HTTPS: Secure all data transmissions with TLS 1.2+.
- Restrict CORS policy: Limit cross-origin access to trusted domains.
- Implement server-side CAPTCHA validation: Prevent automated abuse.
- Add security headers: X-Frame-Options, CSP, X-Content-Type-Options, HSTS.
- Remove tech stack disclosures: Disable x-powered-by header in Express.

4. Positive Security Measures

Listed below are observations we made while conducting the vulnerability assessment. These are intended to be aspects that show improvement after the previous attack.

- OAuth2 Redirect URI Hardening: Strict validation prevents token interception and malicious redirects.
- Strong Authentication Architecture: Google OAuth + Passport.js handles credentials securely, eliminating password storage on servers.
- Role-Based Access Control (RBAC): Access to MongoDB is restricted to internal project accounts, reducing privilege escalation risks.
- CI/CD Security Practices: GitHub Actions ensures automated testing and quality checks.

5. Secure Coding Standards

The development team follows the OWASP Secure Coding Practices Quick Reference Guide, which provides comprehensive guidelines on:

- Input validation and sanitization to prevent injection attacks (e.g., XSS, SQLi).
- Authentication and session management that ensures secure handling of OAuth tokens and JWTs.
- Access control enforcement through Role-Based Access Control (RBAC), restricting privileges based on user roles.
- Error handling and logging that avoids exposing sensitive information while maintaining audit trails.
- Data protection via encryption (AES-256 for data at rest, TLS 1.2+ for data in transit).
- Configuration management to secure default settings and prevent unnecessary disclosures (e.g., stack headers).

The coding standards also include internal development guidelines for secure API design, proper usage of third-party dependencies, and secure handling of user-generated content to prevent abuse.

Code Verification Techniques

To ensure adherence to these secure coding standards, ReliefConnect implements multiple layers of code verification, including automated and manual processes:

Static Application Security Testing (SAST):

SAST tools are integrated into the CI/CD pipeline to automatically scan the codebase for security vulnerabilities, insecure coding patterns, and potential bugs. This enables early detection of issues before code reaches production.

Dependency Scanning:

All third-party libraries and dependencies are monitored using tools like GitHub Dependabot, which identifies known vulnerabilities (CVEs) and suggests updates or patches. This reduces the risk of supply chain attacks.

Linters and Code Quality Tools:

Tools like ESLint and Prettier enforce consistent coding styles, catch syntactic errors, and promote maintainable code, contributing to overall codebase security.

Manual Code Reviews:

All code changes undergo peer-reviewed pull requests, where security-sensitive areas (e.g., authentication flows, data handling logic) receive special attention. Reviewers assess compliance with secure coding standards, logic correctness, and potential security implications.

CI/CD Pipeline Security:

The CI/CD process, powered by GitHub Actions, includes automated testing for functionality, security, and quality. This ensures that only code that passes these checks is deployed to production.

6. Security Requirements Traceability Matrix (SRTM)

This matrix outlines key software requirements for ReliefConnect and their associated security controls, ensuring comprehensive coverage across Confidentiality, Integrity, Availability, Authentication, Authorization, and Auditing (CIA-AAA).

| # | Software Requirement | Security Controls | Categories |
|---|------------------------------------|--|--|
| 1 | Google OAuth Authentication | Enforce strict redirect URI validation; Secure token storage (HTTP-only, Secure) | Authentication, Confidentiality |
| 2 | CAPTCHA on Registration/Login | Server-side CAPTCHA validation; Google reCAPTCHA v2+ | Authentication, Availability |
| 3 | HTTPS Enforcement on All Endpoints | TLS 1.2+, HTTPS redirection, HSTS header | Confidentiality, Integrity, Availability |
| 4 | Sensitive Data Storage | AES-256 encryption; MongoDB encryption at rest; RBAC for key management | Confidentiality, Integrity |
| 5 | CORS Policy | Restrict origins; Limit methods/headers | Authorization, Integrity |
| 6 | Session Management | JWT tokens with expiry; Auto-logout; Invalidate on logout | Authentication, Confidentiality, Integrity |
| 7 | Role-Based Access Control (RBAC) | App and DB-level RBAC; Separate user/admin roles | Authorization, Confidentiality |

7. Secure Design Principles

| # | Design Principle | How should this principle be applied to ReliefConnect? |
|---|--------------------|---|
| 1 | Least Privilege | Implement Role-Based Access Control (RBAC) to ensure that each user, whether a general user, administrator, or developer, has only the permissions necessary for their tasks. Regular users should only access their own data and API functions, while database write/read permissions are limited to internal service accounts. This minimizes potential misuse or accidental data exposure. |
| 2 | Defense in Depth | Apply multiple layers of security controls at different levels of the system. Use HTTPS/TLS for encrypting data in transit, AES-256 encryption for data at rest, CAPTCHA with server-side validation to prevent automated abuse, security headers (e.g., CSP, X-Frame-Options) to secure browser interactions, and RBAC for internal system controls. This ensures that if one control fails, others remain in place to mitigate risks. |
| 3 | Secure by Default | Configure ReliefConnect with security features enabled by default. This includes enforcing HTTPS for all communications, restricting CORS policies to trusted domains, using secure cookie flags (HTTP-only, Secure) for session tokens, and enabling security headers by default. Developers or administrators must take explicit actions to weaken these protections, ensuring the system starts secure. |
| 4 | Fail-Safe Defaults | Design the system to fail securely in case of errors or unexpected conditions. For example, if OAuth2 redirect validation fails, access is denied by default. If the CAPTCHA validation service is unavailable, submissions are blocked rather than allowed. Error messages shown to users are generic (“An error occurred”), while detailed logs are kept internally, preventing sensitive information exposure. |

| | | |
|---|----------------------------|---|
| 5 | Secure Communication | Encrypt all communication channels between clients and servers using HTTPS/TLS. Implement HTTP Strict Transport Security (HSTS) to prevent downgrade attacks. All sensitive data, including user PII and session tokens, should only be transmitted over secure channels. This ensures confidentiality and integrity of data during transit, especially in disaster scenarios where users might access the app over unsecured networks. |
| 6 | Security Through Obscurity | Do not rely solely on hiding system details for security. Instead, use industry-standard, tested security controls like OAuth2, AES-256 encryption, JWT session tokens, and reCAPTCHA. While removing stack disclosure headers (e.g., <code>x-powered-by</code>) limits attacker reconnaissance, core security relies on robust, transparent mechanisms that can withstand scrutiny. |

Data Protection Methods

All data protection mechanisms within ReliefConnect are implemented through trusted third-party service providers, specifically MongoDB Atlas and the Google Identity Platform. Rather than performing cryptographic operations or managing authentication flows directly, these platforms handle the responsibilities, both of which enforce industry-standard security practices and hold recognized compliance certifications such as SOC 2, ISO 27001, and GDPR. Data storage and encryption are handled entirely by MongoDB Atlas, which provides AES-256 encryption at rest enabled by default and TLS/SSL encryption in transit for all database connections. On the authentication side, ReliefConnect uses OAuth 2.0 authorization flows provided through Google Identity Platform to securely verify user identity and manage session state via signed JSON Web Tokens (JWTs). Credential storage and token issuance are managed exclusively by Google's infrastructure, ensuring that the application itself does not directly handle passwords or sensitive authentication data. By relying on these established platforms for both data protection and user identity management, our project significantly reduces its attack surface while aligning with best practices for security and encryption.

8. Testing Methodology

When conducting vulnerability assessments, it is important to adhere to a methodology. Through our assessment, we utilized the Penetration Testing Execution Standard (PTES) framework to model the engagement.

- Pre-engagement Interactions - Defining scope and Rules of Engagement (RoE).
- Intelligence Gathering - Collecting OSINT and researching related technologies.
- Threat Modeling - Identifying business-critical assets that a threat actor may target.
- Vulnerability Analysis - Performing surface level scans to find potential threat vectors.
- Exploitation - Leveraging threat vectors to gain access to target systems.
- Post-Exploitation - Escalate privileges, exfiltrate data, and pivot to internal infrastructure.
- Reporting - Disclosing discovered vulnerabilities, their risk level, and remediation techniques.

9. Risk Assessment Methodology

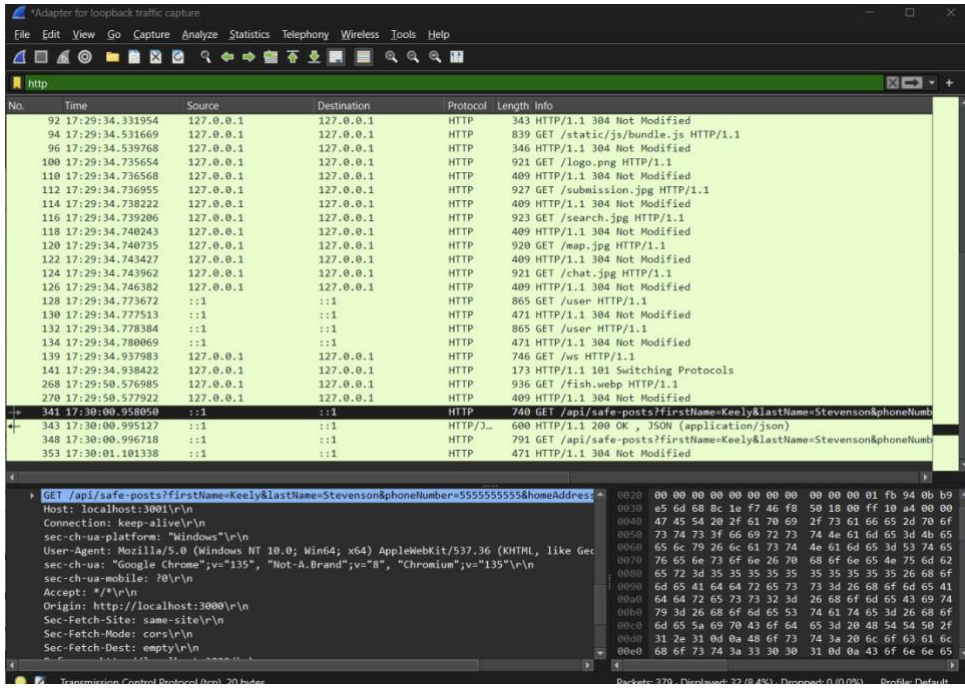
The assessment findings in this report follow the Common Vulnerability Scoring System (CVSS) v3.1 to evaluate the severity of each vulnerability. The CVSS scoring system includes base vulnerability factors as well as temporal and environmental factors. Business impact is further explained in the technical details.

| Severity | CVSS v3.1 Score |
|---------------|-----------------|
| Informational | 0.0 |
| Low | 0.1-3.9 |
| Moderate | 4.0-6.9 |
| High | 7.0-8.9 |
| Critical | 9.0-10.0 |

10. Assessment Findings

Finding 1

| | | | |
|---|--|-------|-----|
| Critical | Cleartext Transmission of Sensitive Data (HTTP) | | |
| Common Vulnerability Scoring System (CVSS) v3.1 | | | |
| Severity | Critical | Score | 9.1 |
| Vector | AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:H/A:N | | |
| Technical Details | | | |
| Affected Systems | ReliefConnect Web Application & APIs | | |
| Description | Sensitive data (e.g., names, phone numbers) is transmitted over unencrypted HTTP, exposing it to Man-in-the-Middle (MITM) attacks. | | |
| Business Impact | <ul style="list-style-type: none">PII exposure via MITM attacks.Regulatory violations (e.g., GDPR, CCPA).Severe reputational damage. | | |
| Remediation | <ul style="list-style-type: none">Enforce HTTPS site-wide.Implement HSTS to prevent protocol downgrades. | | |
| Steps to Reproduce | <ol style="list-style-type: none">Use Wireshark or tcpdump to monitor local network traffic.Submit a form on http://localhost:3000 (e.g., Safe & Well registration).Verify that sensitive data (PII) (e.g., names, phone numbers) is transmitted in cleartext. | | |

| | |
|------------|--|
| |  |
| References | <ol style="list-style-type: none"> https://cwe.mitre.org/data/definitions/319.html https://cheatsheetseries.owasp.org/IndexASVS.html#v91-communications-security-requirements |

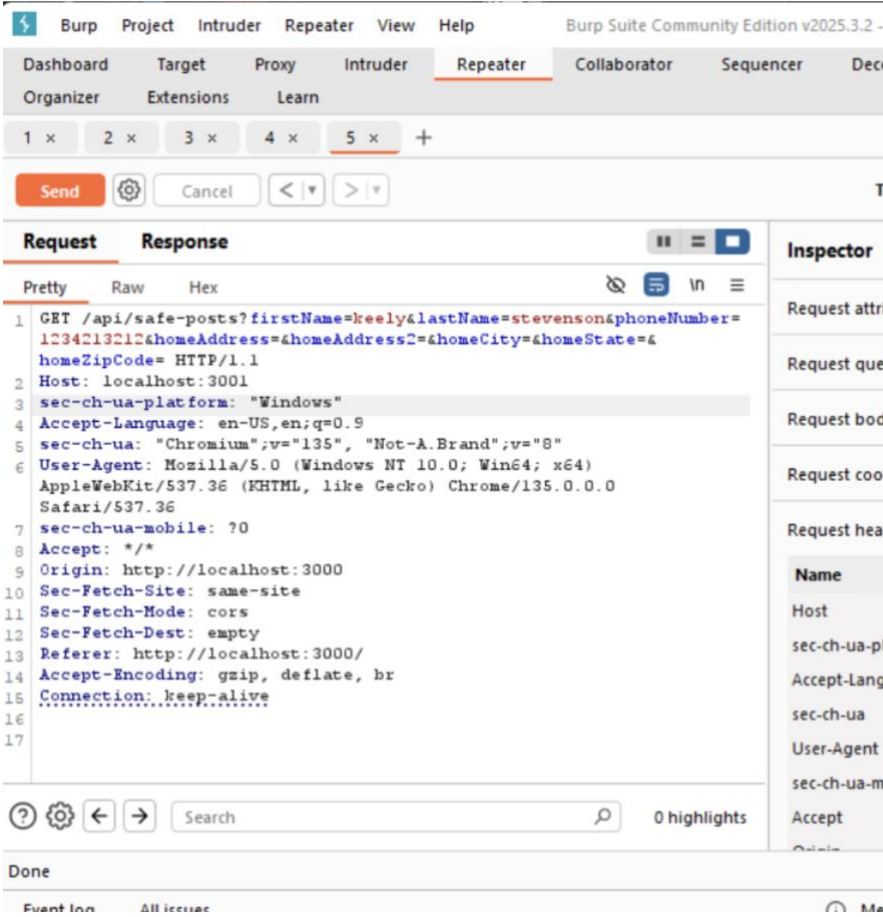
Finding 2

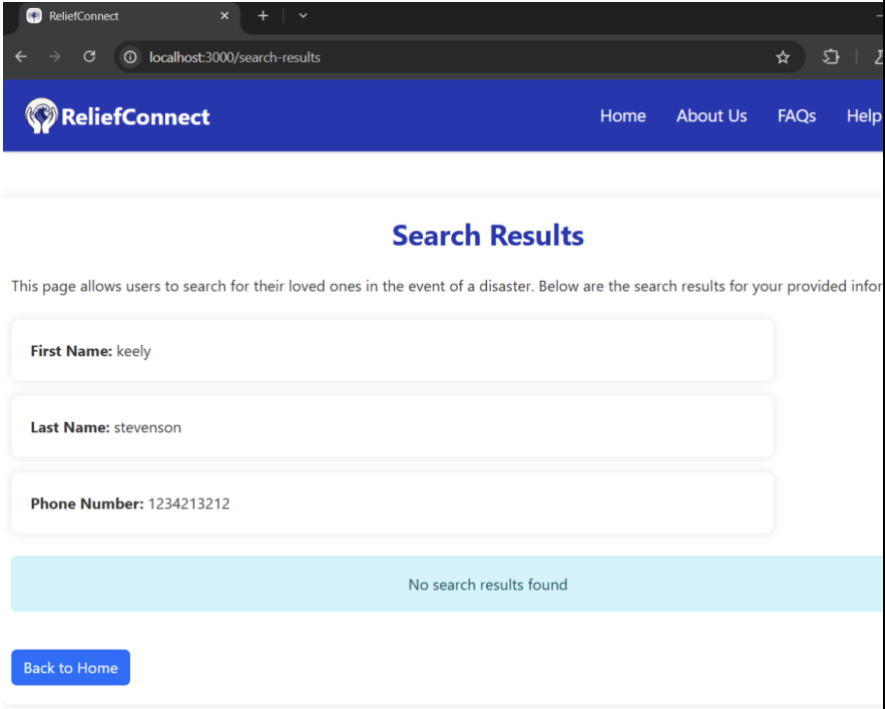
| | | | |
|---|---|-------|-----|
| High | Permissive CORS Policy | | |
| Common Vulnerability Scoring System (CVSS) v3.1 | | | |
| Severity | High | Score | 8.8 |
| Vector | AV:N/AC:L/PR:N/UI:R/S:C/C:H/I:H/A:N | | |
| Technical Details | | | |
| Affected Systems | ReliefConnect API Endpoints | | |
| Description | The Access-Control-Allow-Origin header is set to *, allowing any domain to interact with the ReliefConnect API. This permissiveness introduces cross-origin risks such as CSRF or unauthorized access to session data | | |

| | |
|---------------------------|--|
| Business Impact | <ul style="list-style-type: none"> • Data theft via malicious websites. • Potential for CSRF attacks, allowing attackers to perform actions on behalf of authenticated users. • Risk of integrity and confidentiality breaches. |
| Remediation | <ul style="list-style-type: none"> • Restrict Access-Control-Allow-Origin to trusted domains. • Limit allowed methods (e.g., GET, POST). • Disable wildcard settings in production. |
| Steps to Reproduce | <pre>curl -I http://localhost:3000/api/resources</pre> <ol style="list-style-type: none"> 1. Observe the **Access-Control-Allow-Origin: * ** header in the response. 2. Confirm that any domain is allowed to access the API. |
| References | <ol style="list-style-type: none"> 1. https://cwe.mitre.org/data/definitions/942.html 2. https://owasp.org/API-Security/editions/2023/en/oxa4-unrestricted-resource-consumption/ |

Finding 3

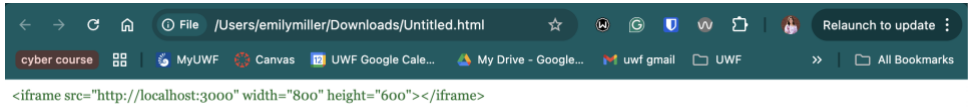
| | | | |
|---|---|-------|-----|
| High | Captcha Not Enforced Server-Side | | |
| Common Vulnerability Scoring System (CVSS) v3.1 | | | |
| Severity | High | Score | 7.0 |
| Vector | AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:L/A:L | | |
| Technical Details | | | |
| Affected Systems | ReliefConnect Registration and Login Forms | | |
| Description | CAPTCHA validation is only implemented client-side. Submissions bypass CAPTCHA by directly sending crafted HTTP requests, enabling automated abuse (e.g., brute force, spam). | | |
| Business Impact | <ul style="list-style-type: none">● Risk of Denial of Service (DoS) via bot-driven fake registrations or logins.● Degradation of availability and potential resource exhaustion. | | |
| Remediation | <ul style="list-style-type: none">● Implement server-side CAPTCHA validation (e.g., Google reCAPTCHA v2/v3).● Reject any requests that bypass CAPTCHA without proper | | |

| | |
|---------------------------|---|
| | validation. |
| Steps to Reproduce | <div><div><div>1. Intercept a registration or login request using Burp Suite or similar tools while solving CAPTCHA.</div><div></div><div><div>2. Resubmit the intercepted request to http://localhost:3000 without solving CAPTCHA or with an invalid CAPTCHA token.</div><div>3. Confirm that the request is processed successfully, bypassing CAPTCHA</div></div></div></div> |

| | |
|-------------------|--|
| |  |
| References | 1. https://cwe.mitre.org/data/definitions/693.html |

Finding 4

| | | | |
|---|---|-------|-----|
| Moderate | Missing Clickjacking Protection (X-Frame-Options) | | |
| Common Vulnerability Scoring System (CVSS) v3.1 | | | |
| Severity | Moderate | Score | 4.3 |
| Vector | AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N | | |
| Technical Details | | | |
| Affected Systems | ReliefConnect Web Application | | |
| Description | The X-Frame-Options header is missing, allowing the app to be embedded in iframes. This opens the door to clickjacking attacks, where malicious sites can trick users into performing unintended actions. | | |

| | |
|---------------------------|---|
| Business Impact | <ul style="list-style-type: none"> • Risk of unauthorized actions by users under false pretenses. • Data integrity compromise. |
| Remediation | <ul style="list-style-type: none"> • Add X-Frame-Options: DENY or SAMEORIGIN header. |
| Steps to Reproduce | <ol style="list-style-type: none"> 1. Create a simple HTML file on any machine: <pre><iframe src="http://localhost:3000" width="800" height="600"></iframe></pre> 2. Open the HTML file in a browser. 3. Verify that http://localhost:3000 renders inside the iframe without restrictions  <p>The screenshot shows a web browser window with the address bar displaying the file path: /Users/emilymiller/Downloads/Untitled.html. The browser's developer tools or address bar shows the iframe src attribute: <iframe src="http://localhost:3000" width="800" height="600"></iframe>. The browser's taskbar at the bottom shows various open applications including 'cyber course', 'MyUWF', 'Canvas', 'UWF Google Cale...', 'My Drive - Google...', 'uwf gmail', and 'UWF'.</p> |
| References | <ol style="list-style-type: none"> 1. https://cwe.mitre.org/data/definitions/1021.html 2. https://github.com/OWASP/ASVS/blob/master/4.0/en/ox22-V14-Config.md |

Finding 5

| | | | |
|---|---|-------|-----|
| Moderate | Missing MIME Sniffing Protection (X-Content-Type-Options) | | |
| Common Vulnerability Scoring System (CVSS) v3.1 | | | |
| Severity | Moderate | Score | 4.3 |
| Vector | AV:N/AC:L/PR:N/UI:R/S:U/C:L/I:L/A:N | | |
| Technical Details | | | |
| Affected Systems | ReliefConnect Web Application | | |

| | |
|---------------------------|---|
| Description | The X-Content-Type-Options: nosniff header is missing. This allows browsers to perform MIME sniffing, which could lead to XSS or incorrect content rendering. |
| Business Impact | <ul style="list-style-type: none">• Risk of XSS or improper data interpretation.• Data integrity issues. |
| Remediation | <ul style="list-style-type: none">• Add X-Content-Type-Options: nosniff header to all responses. |
| Steps to Reproduce | <pre>curl -I http://localhost:3000</pre> <ol style="list-style-type: none">1. Check the response headers.2. Confirm that X-Content-Type-Options: nosniff is missing. <pre>emilymiller@Emilys-MacBook-Pro ~ % curl -I http://localhost:3000 HTTP/1.1 200 OK X-Powered-By: Express Access-Control-Allow-Origin: * Access-Control-Allow-Methods: * Access-Control-Allow-Headers: * Content-Type: text/html; charset=utf-8 Accept-Ranges: bytes Content-Length: 855 ETag: W/"357-vkiaXut/T0LCcnTbjosuFRtI5tA" Vary: Accept-Encoding Date: Fri, 25 Apr 2025 03:34:15 GMT Connection: keep-alive Keep-Alive: timeout=5</pre> |
| References | <ol style="list-style-type: none">1. https://cwe.mitre.org/data/definitions/943.html |

Finding 6

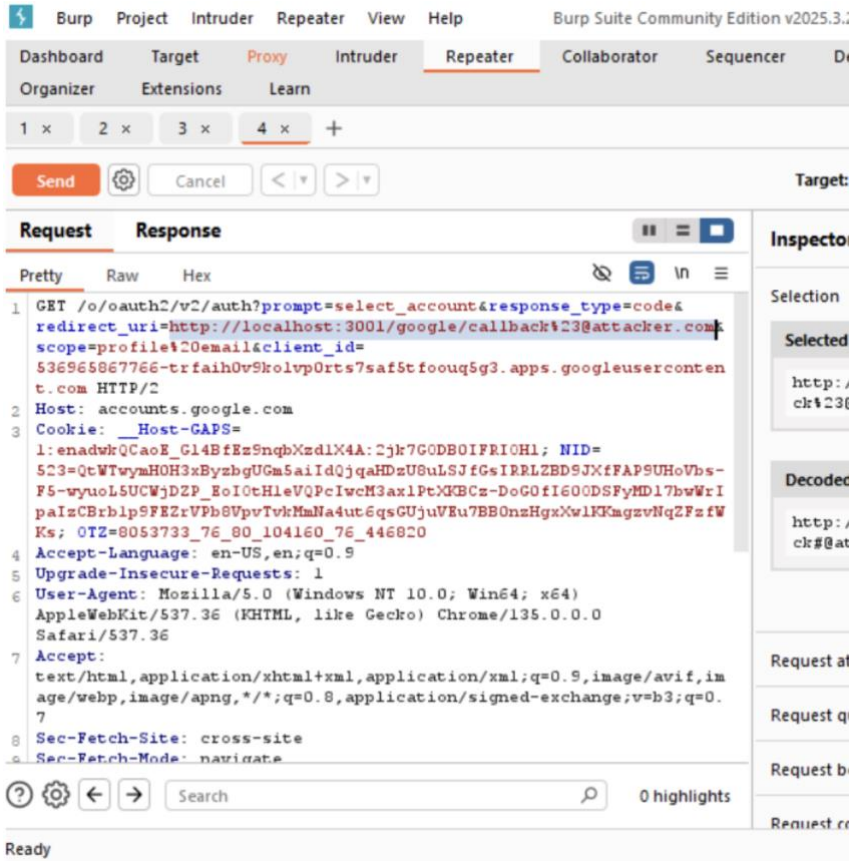
| | | | |
|---|---|-------|-----|
| Low | Tech Stack Disclosure (X-Powered-By: Express) | | |
| Common Vulnerability Scoring System (CVSS) v3.1 | | | |
| Severity | Low | Score | 3.7 |
| Vector | AV:N/AC:L/PR:N/UI:N/S:U/C:L/I:N/A:N | | |
| Technical Details | | | |

| | |
|---------------------------|---|
| Affected Systems | ReliefConnect Web Server |
| Description | The x-powered-by header reveals that the backend framework is Express.js, providing potential attackers with information to tailor their attacks. |
| Business Impact | <ul style="list-style-type: none"> Reconnaissance for targeted attacks. Increased attack surface awareness. |
| Remediation | <ul style="list-style-type: none"> Remove the x-powered-by header in Express.js. |
| Steps to Reproduce | <pre>curl -I http://localhost:3000</pre> <ol style="list-style-type: none"> Look for the x-powered-by: Express header in the HTTP response. Confirm that Express.js is disclosed. <pre>emilymiller@Emilys-MacBook-Pro ~ % curl -I http://localhost:3000 HTTP/1.1 200 OK X-Powered-By: Express Access-Control-Allow-Origin: * Access-Control-Allow-Methods: * Access-Control-Allow-Headers: * Content-Type: text/html; charset=utf-8 Accept-Ranges: bytes Content-Length: 855 ETag: W/"357-vkiaXut/T0LCcnTbjosuFRtI5tA" Vary: Accept-Encoding Date: Fri, 25 Apr 2025 03:34:15 GMT Connection: keep-alive Keep-Alive: timeout=5</pre> |
| References | <ol style="list-style-type: none"> https://cwe.mitre.org/data/definitions/200.html |

Secure Finding 1

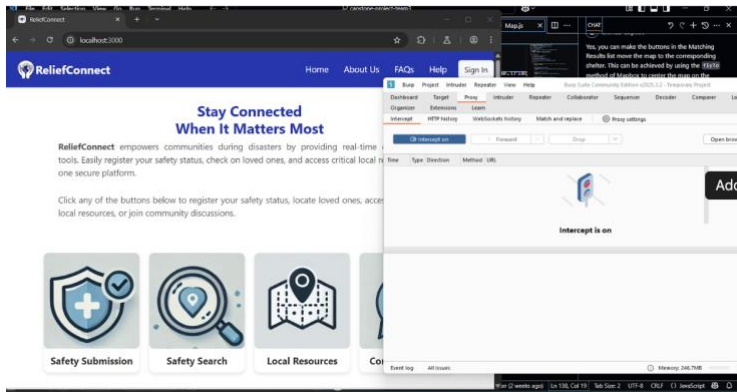
| | | | |
|---|----------------------------------|-------|-----|
| None | OAuth2 Redirect URI Manipulation | | |
| Common Vulnerability Scoring System (CVSS) v3.1 | | | |
| Severity | None | Score | 0.0 |
| Vector | N/A | | |

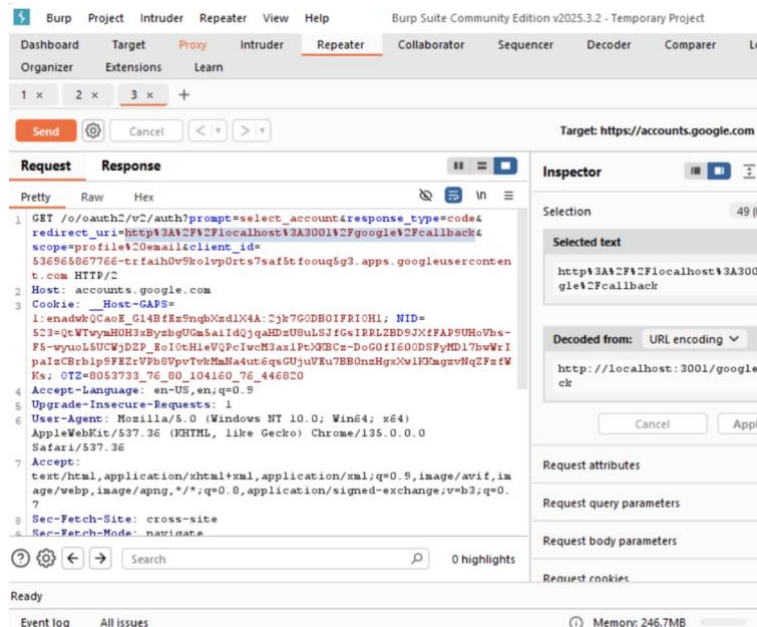
| Technical Details | |
|---------------------------|---|
| Affected Systems | ReliefConnect Google OAuth2 Login Flow |
| Description | <p>During testing, we attempted to manipulate the Google OAuth2 redirect URI by adding encoded characters (e.g., %23, representing #) to test for redirect bypass opportunities. This is a common method in redirect attacks where attackers try to inject unauthorized domains after a fragment identifier to trick the system into redirecting to a malicious site.</p> <p>The ReliefConnect application and Google OAuth2 handled this securely. The system correctly treated the URI portion before the fragment (#) as the valid redirect URI, and ignored the injected content. Unauthorized domains embedded after the fragment were not accepted.</p> |
| Business Impact | <ul style="list-style-type: none"> None |
| Remediation | <ul style="list-style-type: none"> None |
| Steps to Reproduce | <ol style="list-style-type: none"> Initiate a Google OAuth2 login request from http://localhost:3000. Intercept the request (using Burp Suite). |

| | |
|------------|---|
| | <div><p>The screenshot shows the Burp Suite interface with the 'Repeater' tab selected. A GET request is displayed in the 'Request' pane, with the 'redirect_uri' parameter manipulated to include a fragment injection: <code>redirect_uri=http://localhost:3001/google/callback#23@attacker.com</code>. The 'Inspector' pane on the right shows the selected request. The status bar at the bottom indicates 'Ready'.</p></div> <div><ol style="list-style-type: none">3. Modify the redirect_uri parameter to include a fragment injection: https://trustedsite.com#@malicious.com4. Observe that Google OAuth2 rejects the manipulated redirect URI, preventing redirection to unauthorized domains.</div> |
| References | <ol style="list-style-type: none">1. https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html |

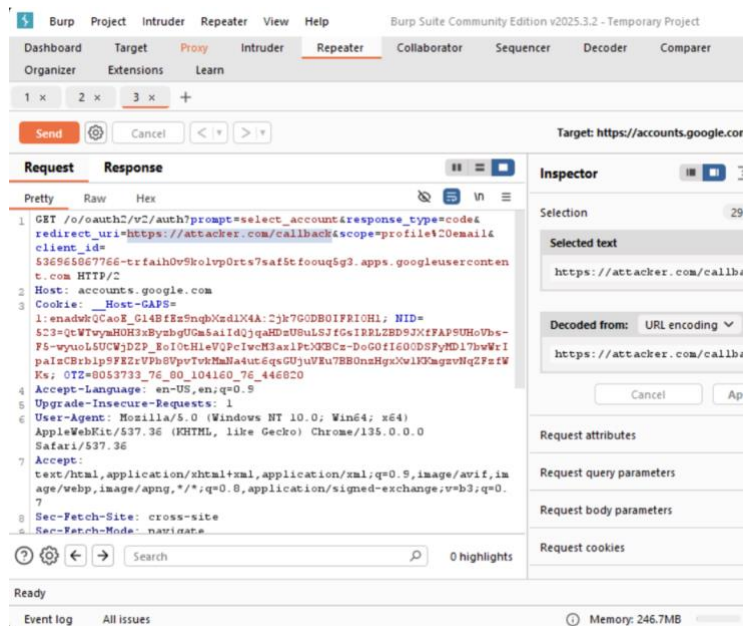
Secure Finding 2

| | | | |
|---|---|-------|-----|
| None | OAuth2 Redirect URI Validation (URL Bypass Attempt) | | |
| Common Vulnerability Scoring System (CVSS) v3.1 | | | |
| Severity | None | Score | 0.0 |
| Vector | N/A | | |

| Technical Details | |
|--------------------|--|
| Affected Systems | ReliefConnect Google OAuth2 Login Flow |
| Description | <p>As part of the penetration testing process, we examined the Google OAuth2 login flow in ReliefConnect to ensure that redirect URIs are securely validated. The redirect URI is the address where Google sends users after successful authentication. If improperly restricted, attackers could manipulate this URI to intercept OAuth tokens and hijack sessions.</p> <p>To test this, we intercepted the login request and altered the redirect_uri parameter to point to unauthorized or attacker-controlled domains. However, Google OAuth2 correctly identified the tampered request and rejected the login attempt, refusing to send tokens to unapproved URLs. The login flow was halted with an error response, preventing token leakage.</p> <p>This confirms that OAuth2 redirect URI validation is properly configured and adheres to best security practices, ensuring the application is secure against URL bypass attacks.</p> |
| Business Impact | <ul style="list-style-type: none">None |
| Remediation | <ul style="list-style-type: none">None |
| Steps to Reproduce | <ol style="list-style-type: none">Initiate the Google OAuth2 login on http://localhost:3000.Intercept the authentication request using Burp Suite or similar tools.  |



3. Modify the redirect_uri parameter to an unauthorized domain (e.g., <https://attacker.com/callback>).



4. Observe that Google OAuth2 rejects the manipulated request and halts the login process with an error message.

| | |
|-------------------|--|
| |  <p>The top screenshot shows the ReliefConnect website with navigation links (Home, About Us, FAQs, Help, Sign In) and a 'Stay Connected When It Matters Most' section. Below this are icons for 'Safety Submission', 'Safety Search', and 'Local Resources'. The bottom screenshot shows the Burp Suite interface with the 'Repeater' tab selected. The target is 'https://accounts.google.com'. The request is a GET to '/signin/oauth/error?authError=...' and the response is a 200 OK from 'accounts.google.com' with a 'Host' cookie.</p> |
| References | <ol style="list-style-type: none">1. https://cheatsheetseries.owasp.org/cheatsheets/Unvalidated_Redirects_and_Forwards_Cheat_Sheet.html2. https://developers.google.com/identity/protocols/oauth2/resources/best-practices |

11. Mitigation Strategy

| Priority | Risk | Mitigation Strategy |
|----------|------|---------------------|
|----------|------|---------------------|

| | | |
|---|---------------------------------------|---|
| 1 | Transmission of Sensitive Data (HTTP) | Enforce HTTPS site-wide. Configure HSTS. |
| 2 | Permissive CORS Policy | Restrict Access-Control-Allow-Origin to trusted domains. Limit methods and headers. |
| 3 | CAPTCHA Not Enforced Server-Side | Implement server-side CAPTCHA validation (reCAPTCHA v2/v3). |
| 4 | Missing Security Headers | Add headers: X-Frame-Options , CSP , X-Content-Type-Options , Strict-Transport-Security . |
| 5 | Technology Stack Disclosure | Remove x-powered-by header in Express.js. |
| 6 | Insufficient Logging and Auditing | Implement tamper-resistant logs for critical actions (e.g., login, data changes). |

12. Remaining Security Issues

| Finding | Severity | Status |
|--------------------------------------|----------|-------------------------|
| Cleartext Transmission (HTTP) | Critical | Remediation in progress |
| Permissive CORS Policy | High | Pending remediation |
| CAPTCHA Not Enforced Server-Side | High | Pending remediation |
| Missing Security Headers | Moderate | Planned remediation |
| Tech Stack Disclosure (X-Powered-By) | Low | Planned remediation |

13. Summary & Sufficiency Justification

The security documentation presented in this report offers a comprehensive evaluation of ReliefConnect's security posture across the application, network, and system layers. The documentation addresses critical security concerns spanning authentication mechanisms, data protection methods, system configurations, and resilience against common and emerging threats.

Coverage Overview:

- Application Layer Security:
 - Threat modeling of the web application and APIs.
 - Secure authentication via Google OAuth and CAPTCHA mechanisms.
 - Secure coding standards with input validation, role-based access control (RBAC), and dependency management.
 - Protection against common vulnerabilities like XSS, CSRF, clickjacking, and API abuse.
- Network Layer Security:
 - Encryption in transit enforced through HTTPS with TLS 1.2+ and HSTS headers.
 - CORS policy management to prevent unauthorized cross-origin access.
 - Threat detection for network-based attacks, including Man-in-the-Middle (MITM) scenarios.
- System Layer Security:
 - Data encryption at rest (AES-256 for MongoDB).
 - CI/CD pipeline security with automated tests, static analysis, and secure deployment practices.
 - Elimination of unnecessary disclosures (e.g., tech stack masking via header removal).

Methodologies Applied:

- Penetration Testing Execution Standard (PTES):
Utilized to guide the vulnerability assessment process, covering pre-engagement interactions, intelligence gathering, threat modeling, vulnerability analysis, exploitation, post-exploitation, and reporting.
- Common Vulnerability Scoring System (CVSS) v3.1:
Applied to assess the severity of discovered vulnerabilities, factoring in both technical risk and business impact.
- Secure Design Principles:
Integrated throughout the system design, including Least Privilege, Defense in Depth, Secure by Default, Fail-Safe Defaults, and Secure Communication. These principles ensure that even if individual controls fail, compensating mechanisms remain active.
- OWASP Standards and Frameworks:
Security practices and threat identification align with the OWASP Top 10, OWASP API Security Top 10, and OWASP Secure Coding Guidelines.

Sufficiency Justification:

The depth and breadth of the current security documentation, combined with the testing methodologies and design principles, provide adequate coverage for the existing risk profile of ReliefConnect. The system's major components have been evaluated against common vulnerabilities, potential abuse cases, and best security practices.

The known vulnerabilities identified during the penetration test (e.g., HTTP transmission of sensitive data, permissive CORS policy, client-side CAPTCHA validation) have been clearly outlined with remediation guidance, ensuring that mitigation can be prioritized based on risk severity. Moreover, critical security controls like OAuth2 hardening, role-based access control, encryption mechanisms, and secure deployment practices are already in place, reducing the system's overall attack surface.

As such, the documentation is sufficient for the current stage of ReliefConnect's lifecycle, addressing high-priority risks while providing a roadmap for continuous improvement through remediation and future assessments.

ReliefConnect has demonstrated a strong commitment to security, with robust controls already in place for authentication, data protection, and infrastructure security. While several vulnerabilities were identified, clear remediation strategies have been outlined and prioritized.

15. Summary

Addressing these issues will further strengthen the platform's resilience, ensuring it remains a reliable resource for users during critical disaster scenarios. Ongoing security assessments and adherence to secure development practices are recommended to maintain and improve the platform's security posture.

Appendix C: Tools & References

Tools Used During Assessment

- Burp Suite– Web app testing, proxying, and vulnerability scanning.
- Wireshark – Network packet analysis for traffic inspection.
- nmap – Network mapping and service enumeration.
- curl – Manual HTTP requests and response header inspection.
- tcpdump – CLI packet capture tool.
- OWASP ZAP – Passive/active vulnerability scanning.
- Metasploit Framework – Used for controlled exploitation testing.
- Dirbuster/Dirsearch – Directory and resource enumeration.
- JWT.io – For analyzing and validating JWT tokens.
- Postman – API request crafting and testing.
- Nikto – Web server scanner for outdated components and misconfigurations.

Standards and Frameworks Referenced

- Penetration Testing Execution Standard (PTES)
- OWASP Application Security Verification Standard (ASVS)
- OWASP API Security Top 10 (2023)
- Common Vulnerability Scoring System (CVSS) v3.1
- Common Weakness Enumeration (CWE)
- NIST SP 800-53
- ISO/IEC 27001