INDEPENDENT SECURITY RESEARCH

CONFIDENTIAL · REDACTED

Security Audit Report

보안 취약점 점검 보고서

TARGET

[redacted].org

REPOSITORY

github.com/[peer]/[redacted]

FIREBASE PROJECT

[firebase-project-redacted]

RESEARCHER

Geonhee (건희)

REPORT DATE

April 18, 2026

FINDINGS

15 (3C · 3H · 4M · 4L · 1I)

Table of Contents · 목차

REDACTION NOTICE. This is a public-release version of an internal security audit. Target domain, repository owner, project identifiers, and credentials have been replaced with redacted placeholders. The original full report was disclosed privately to the repository owner. Findings, methodology, and remediation guidance are unchanged.

공개 안내. 본 문서는 비공개 감사 보고서의 공개용 redacted 버전입니다. 대상 도메인·리포지토리 소유자·프로젝트 식별자·자격 증명은 placeholder로 가림 처리되었으며, 발견 사항·방법론·조치 가이드는 원본 그대로입니다.

1. Executive Summary

EN This report documents an independent security audit conducted against `[redacted].org`, a student-run school community website for Chadwick International ("Sixers"). The assessment was performed with the repository owner's consent, established by his public disclosure of the source repository to the entire school community. The audit covered the live frontend, its public GitHub repository, and inferred backend configuration.

EN The site exhibits a classic "static-frontend + Firebase BaaS" architecture common among student projects. The audit identified 15 findings across the full severity spectrum. Three findings are classified as Critical and require immediate action — most notably the accidental disclosure of a Google OAuth 2.0 Client Secret in a committed `.env` file, which enables brand-impersonation phishing against school-domain identities and must be revoked at once. Authentication enforcement currently relies entirely on client-side JavaScript checks, which provide no meaningful security boundary. The state of server-side Firestore and Storage security rules could not be externally verified and may represent an additional Critical issue depending on their configuration.

EN The combination of exposed OAuth credentials, weak identity verification, and potentially permissive database rules means that — under conservative assumptions — a motivated attacker could impersonate the school's OAuth application, harvest `@chadwickschool.org` access tokens, and read or modify all community data stored in Firestore. Remediation guidance, including code snippets and rule templates, is provided for each finding.

> KR 이 보고서는 Chadwick International 재학생들이 운영하는 학내 커뮤니티 웹사이트 `[redacted].org`에 대한 독립적 보안 점검 결과입니다. 점검은 리포지토리 소유자가 학교 공지를 통해 소스코드 주소를 전체 공개한 사실을 근거로 수행되었으며, 라이브 프론트엔드, 공개된 GitHub 리포지토리, 그리고 유추 가능한 백엔드 구성 전반을 대상으로 했습니다.
>
> KR 해당 사이트는 학생 프로젝트에서 흔히 볼 수 있는 "정적 프론트엔드 + Firebase BaaS" 구조입니다. 점검 결과 총 15건의 취약점이 식별되었고, 그중 Critical 등급 3건은 즉각적인 대응이 필요합니다. 특히 리포지토리에 커밋된 `.env` 파일에서 Google OAuth 2.0 Client Secret이 그대로 노출된 점이 가장 심각합니다. 이 자격증명은 학교 도메인 사용자들을 상대로 한 브랜드 사칭 피싱에 악용될 수 있으며, 즉시 폐기/재발급이 필요합니다. 또한 인증 강제 로직이 전적으로 클라이언트 사이드 JavaScript 검증에 의존하고 있어 보안적 의미가 없는 상태이며, 서버 사이드 Firestore·Storage 보안 규칙의 상태는 외부에서 확인 불가능하기 때문에 설정에 따라 추가 Critical 이슈가 될 가능성이 있습니다.
>
> KR 노출된 OAuth 자격증명, 허술한 사용자 신원 검증, 그리고 잠재적으로 과도하게 허용된 DB 규칙이 결합되면, 공격자가 학교 OAuth 애플리케이션을 사칭해 `@chadwickschool.org` 계정의 액세스 토큰을 획득하고 Firestore에 저장된 모든 커뮤니티 데이터를 열람·조작할 수 있는 위험이 존재합니다. 각 취약점별 조치 방안과 수정 코드 템플릿을 본문에 포함했습니다.

2. Scope & Methodology

IN-SCOPE ASSETS

ASSET	TYPE	EXPOSURE
`[redacted].org`	Static site (GitHub Pages)	Public
`github.com/[peer]/[redacted]`	Source repository	Public
	Upstream fork parent	Public (referenced only)

ASSET	TYPE	EXPOSURE
`github.com/[upstream-author]/[upstream-repo]`		
Firebase project `[firebase-project-redacted]`	Backend (Auth, Firestore, Storage)	Inferred from client SDK config
Google Cloud project `[gcp-project-redacted]`	OAuth 2.0 client	Inferred from `.env`

METHODOLOGY

EN The audit was passive and non-invasive. No exploitation was performed against live user data; all identified issues were confirmed through (a) public source-code review, (b) inspection of HTTP responses and response headers, (c) reading committed files in the GitHub repository including git history, and (d) logical analysis of client-side authentication flows. No fuzzing, brute force, denial of service, or social engineering was attempted. No attempt was made to authenticate or query backend services beyond a single anonymous metadata probe to characterize exposure.

> KR 본 점검은 수동(passive) 및 비침투적(non-invasive) 방식으로 수행되었습니다. 실제 사용자 데이터에 대한 공격 실행은 일절 이루어지지 않았으며, 모든 발견 사항은 ① 공개 소스코드 리뷰, ② HTTP 응답 및 헤더 분석, ③ GitHub 리포지토리의 커밋된 파일과 히스토리 열람, ④ 클라이언트 사이드 인증 흐름의 논리적 분석을 통해 확인되었습니다. 퍼징, 무차별 대입, 서비스 거부(DoS), 사회공학(social engineering) 공격은 시도하지 않았으며, 백엔드 서비스에 대한 인증·쿼리 역시 노출 상태 확인을 위한 익명 메타데이터 프로브 1건을 제외하고는 수행하지 않았습니다.

SEVERITY CLASSIFICATION

Severity ratings follow CVSS 3.1. Values reflect the researcher's assessment based on observable evidence; some Critical/High ratings are conditional on backend configuration states that were not externally confirmable at the time of the report.

3. Findings Summary

ID	TITLE	SEVERITY	CVSS
CISIXERS-001	Exposed Google OAuth Client Secret in committed .env	CRITICAL	9.1
CISIXERS-002	Firestore Security Rules state unverified (potentially permissive)	CRITICAL	9.8*
CISIXERS-003	Authentication enforcement is entirely client-side	CRITICAL	8.1
CISIXERS-004	Race condition in requireAuth() permits pre-redirect data exposure	HIGH	7.5
CISIXERS-005	Weak email-domain validation; missing Google hd parameter	HIGH	7.1
CISIXERS-006	Firebase Storage rules state unverified	HIGH	6.8*
CISIXERS-007	Sensitive user data cached in localStorage (XSS amplifier)	MEDIUM	6.1
CISIXERS-008	Default SESSION_SECRET placeholder committed	MEDIUM	5.9
CISIXERS-009	Missing security headers (CSP, HSTS, X-Frame-Options, etc.)	MEDIUM	5.4
CISIXERS-010	Missing email_verified claim check in auth flow	MEDIUM	5.3

ID	TITLE	SEVERITY	CVSS
CISIXERS-011	Missing Subresource Integrity (SRI) on CDN resources	LOW	4.3
CISIXERS-012	Raw Firebase error messages exposed to users	LOW	3.7
CISIXERS-013	.DS_Store committed; reveals filesystem structure	LOW	3.1
CISIXERS-014	OAuth callback URL uses insecure http://	LOW	3.1
CISIXERS-015	Repository URL publicly disclosed via school announcement	INFO	–

* Conditional rating — exact severity depends on backend configuration that could not be externally verified.

4. Detailed Findings

CISIXERS-001

Exposed Google OAuth Client Secret in committed .env

CRITICAL | CVSS 3.1 / 9.1 | AV:N/AC:L/PR:N/UI:N/S:C/C:H/I:H/A:N | CWE-798 | CWE-312 | CWE-540

SUMMARY

EN A Google OAuth 2.0 Client Secret is committed in plaintext in `.env` at the repository root. The file is world-readable via GitHub. Any Google OAuth credential beginning with `GOCSPX-` is the confidential half of a client credential pair and must never be publicly exposed.

> KR Google OAuth 2.0 Client Secret이 리포지토리 루트의 `.env` 파일에 평문으로 커밋되어 있으며, 해당 파일은 GitHub를 통해 전 세계 누구나 열람 가능한 상태입니다. `GOCSPX-` 로 시작하는 값은 OAuth 클라이언트 자격증명 쌍의 비밀 부분이며, 절대로 공개되어서는 안 됩니다.

EVIDENCE

Path: `.env` at `github.com/[peer]/[redacted]`

```
GOOGLE_CLIENT_ID=[client-id-redacted].apps.googleusercontent.com
GOOGLE_CLIENT_SECRET=[secret-rotated]
CALLBACK_URL=http://[redacted].org/auth/google/callback
SESSION_SECRET=your_session_secret_key_here
```

IMPACT

EN Even though the production frontend uses the Firebase popup flow (which does not consume the client secret), the exposed secret remains valid in Google Cloud. An attacker can:

- Create a high-fidelity phishing site that presents a genuine Google consent screen under the legitimate client ID, harvesting access tokens from `@chadwickschool.org` users.
- Complete the server-side authorization-code flow to exchange intercepted codes for access and refresh tokens.
- Consume API quota assigned to the project `[gcp-project-redacted]`, potentially incurring cost on the owner's Google Cloud billing account.
- Host a look-alike application under the legitimate school-branded OAuth name, damaging trust.

> KR 프로덕션 프론트엔드가 Firebase 팝업 방식(클라이언트 시크릿을 사용하지 않음)을 쓰고 있더라도, 노출된 시크릿은 Google Cloud 상에서 여전히 유효합니다. 공격자는 동일한 Client ID로 정상처럼 보이는 Google 동의 화면을 띄우는 피싱 사이트를 구축해 `@chadwickschool.org` 사용자의 액세스 토큰을 탈취하거나, 서버 사이드 authorization code flow를 완성하여 가로챈 code를 access/refresh token으로 교환하거나, 프로젝트 `[gcp-project-redacted]`의 API 쿼터를 소진시켜 소유자의 Google Cloud 결제 계정에 비용을 발생시키거나, 학교 브랜드 명의의 OAuth 앱 이름으로 유사 애플리케이션을 호스팅하여 신뢰를 훼손할 수 있습니다.

REMEDIATION (IMMEDIATE)

1. Revoke and rotate the client secret in Google Cloud Console → APIs & Services → Credentials → OAuth 2.0 Client IDs → "Reset secret". This invalidates the leaked value globally.
2. Remove `.env` from the repository and add it to `.gitignore`.

3. Purge the file from git history using `git filter-repo` or `bfg --delete-files .env`, followed by force-push. Note: rotation in step 1 must precede this; history purging alone does not invalidate the leaked secret because archives, forks, and caches persist.
4. Audit Google Cloud audit logs for suspicious OAuth token exchanges or API activity during the exposure window.
5. Review the upstream fork `[upstream-author]/[upstream-repo]`; if the `.env` file originated there, notify the upstream owner.

```
# Remediation commands
bfg --delete-files .env
git reflog expire --expire=now --all
git gc --prune=now --aggressive
git push --force
echo ".env" >> .gitignore
echo ".DS_Store" >> .gitignore
```

REFERENCES

- Google — "Handling Client Secrets" (OAuth 2.0 best practices)
- OWASP Top 10: A07:2021 — Identification and Authentication Failures
- CWE-798: Use of Hard-coded Credentials

CISIXERS-002

Firestore Security Rules state unverified (potentially permissive)

CRITICAL | CVSS 3.1 / 9.8* (conditional) | CWE-284 | CWE-862

SUMMARY

EN The repository does not include a `firestore.rules`, `storage.rules`, or `firebase.json` file. This strongly suggests the backend rules are managed directly in the Firebase Console and are not version-controlled. In the common default state ("Start in test mode"), Firestore allows unrestricted read/write from any client for 30 days. If this default was retained, the entire community's data is readable and writable by unauthenticated attackers worldwide.

> KR 리포지토리에 `firestore.rules`, `storage.rules`, `firebase.json`이 존재하지 않습니다. 이는 백엔드 규칙이 Firebase 콘솔에서 수동으로 관리되고 있으며 버전 관리 대상이 아님을 강하게 시사합니다. Firebase가 프로젝트 생성 시 기본 제공하는 "테스트 모드(Start in test mode)"는 30일간 전 세계 누구에게나 read/write를 허용하며, 만약 이 기본 설정이 그대로 유지되고 있다면 커뮤니티 전체 데이터가 비인증 상태로도 열람·조작 가능한 상태입니다.

VERIFICATION STEPS

EN The owner should verify rules immediately in the Firebase Console under Firestore Database → Rules and Storage → Rules. External verification without data access was intentionally not attempted.

```
// Red flag - "test mode" default
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    match /{document=**} {
      allow read, write: if request.time < timestamp.date(2026, 5, 15);
    }
  }
}
```

IMPACT

If rules are permissive, all of the following are possible without any authentication:

- Full export of all user records, posts, comments, and PII.
- Silent creation, modification, or deletion of any document.
- Mass defacement of the bulletin board and community content.
- Unauthorized publishing under arbitrary author identities.

REMEDIATION

Define and deploy explicit security rules with version control. Below is a minimum-viable policy that matches the current authentication model (Chadwick-domain users only, owner-scoped writes).

```
rules_version = '2';
service cloud.firestore {
  match /databases/{database}/documents {
    function isChadwick() {
      return request.auth != null
        && request.auth.token.email_verified == true
```

```
          && request.auth.token.email.matches('.*@chadwickschool[.]org$');
      }
      function isOwner(uid) {
        return request.auth != null && request.auth.uid == uid;
      }

      match /posts/{postId} {
        allow read: if isChadwick();
        allow create: if isChadwick()
          && request.resource.data.authorUid == request.auth.uid;
        allow update, delete: if isChadwick()
          && isOwner(resource.data.authorUid);
      }

      match /users/{userId} {
        allow read: if isChadwick();
        allow write: if isOwner(userId);
      }

      match /{document=**} {
        allow read, write: if false; // deny by default
      }
    }
  }
```

CISIXERS-003

Authentication enforcement is entirely client-side

CRITICAL | CVSS 3.1 / 8.1 | AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:H/A:N | CWE-602 | CWE-287

SUMMARY

EN In `auth.js`, access control for non-school accounts is enforced *after* a successful Google sign-in by calling `signOut(auth)` from JavaScript. This "sign in, then sign out if invalid" pattern creates no real security boundary: any user can bypass it by patching the `signOut` function at runtime, intercepting the promise, or forging the `localStorage` user object directly.

EVIDENCE

```
// auth.js (excerpt)
if (!email.endsWith('@chadwickschool.org')) {
  showError('❌ Please use your @chadwickschool.org email address.');
  signOut(auth);  // <-- client-side enforcement only
  return;
}
localStorage.setItem('user', JSON.stringify({ ... }));
```

PROOF OF CONCEPT

An attacker opens the login page's DevTools Console and executes either of the following before attempting Google sign-in:

```
// Neutralize the post-login sign-out
window.signOut = () => Promise.resolve();

// Or forge the session directly - any value accepted client-side
localStorage.setItem('user', JSON.stringify({
  uid: 'FAKE', name: 'attacker',
  email: 'attacker@chadwickschool.org', photo: ''
}));
location.reload();
```

> KR `auth.js` 에서는 Google 로그인 성공 이후 JavaScript의 `signOut(auth)` 호출로 학교 도메인이 아닌 계정을 제거하는 방식으로 접근 제어를 수행합니다. "일단 로그인시키고, 유효하지 않으면 로그아웃" 패턴은 실질적 보안 경계가 되지 못합니다. 공격자는 런타임에 `signOut` 함수를 덮어쓰거나, Promise를 가로채거나, `localStorage` 의 user 객체를 직접 위조하여 이 검사를 우회할 수 있습니다.

IMPACT

Any Google account holder worldwide can obtain an authenticated application session and — depending on backend rules (see CISIXERS-002) — access and modify community data. School-domain restriction, the sole advertised access control, is effectively absent.

REMEDIATION

Move enforcement into Firebase Security Rules using `request.auth.token.email`, and additionally restrict Google at the provider level using the `hd` (hosted-domain) parameter so non-school accounts cannot complete the OAuth flow in the first place.

```
const provider = new GoogleAuthProvider();
provider.setCustomParameters({ hd: 'chadwickschool.org' });
// Google Workspace will reject non-matching domains pre-consent.
```

Combine with rule-level enforcement from CISIXERS-002 so that even a successful non-school login reads/writes nothing.

CISIXERS-004

requireAuth() race condition permits pre-redirect data exposure

HIGH | CVSS 3.1 / 7.5 | AV:N/AC:L/PR:N/UI:N/S:U/C:H/I:N/A:N | CWE-362 | CWE-613

SUMMARY

EN `requireAuth()` awaits `onAuthStateChanged` asynchronously before redirecting unauthenticated users to `login.html`. In the interim, protected page HTML has already loaded and embedded scripts can initiate Firestore queries. A user who halts the redirect (`window.stop()` , network tab inspection, Esc key during load) can observe data intended only for authenticated users.

EVIDENCE

```
async function requireAuth() {
  const user = await checkAuth();  // async - hundreds of ms
  if (!user && !window.location.pathname.includes('login.html')) {
    window.location.href = './login.html';  // redirect happens late
  }
}
```

PROOF OF CONCEPT

1. Log out. Open DevTools → Network tab, enable "Preserve log".
2. Navigate directly to a protected page (e.g. `/spotlight.html`).
3. Observe that Firestore/Storage requests are issued before redirect; inspect their responses.
4. Optionally press Esc during load or call `window.stop()` to stay on the page and render cached data.

IMPACT

Client-side redirects provide UX, not security. An attacker with network-level visibility sees anything the SDK fetches. Paired with permissive Firestore rules (CISIXERS-002), this enables anonymous read of protected collections.

REMEDIATION

Hide page content until authentication is confirmed, and rely on server-side rules to deny unauthenticated requests regardless of frontend state.

```
<script type="module">
import { getAuth, onAuthStateChanged }
  from "https://www.gstatic.com/firebasejs/12.10.0/firebase-auth.js";

document.documentElement.style.visibility = 'hidden';
onAuthStateChanged(getAuth(), user => {
  const ok = user && /@chadwickschool\.org$/.test(user.email);
  if (!ok) window.location.replace('./login.html');
  else document.documentElement.style.visibility = 'visible';
});
</script>
```

> KR `requireAuth()` 가 `onAuthStateChanged` 를 비동기로 대기한 뒤에 리다이렉트하기 때문에, 대기 시간 동안 보호 대상 페이지의 HTML은 이미 로드되어 있고 내장 스크립트가 Firestore 쿼리를 시작할 수 있습니다. 사용자가 리다이렉트를 중단(`window.stop()` , 네트워크 패널 검사, 로딩 중 Esc)하면 인증된 사용자 전

용 데이터를 관찰할 수 있습니다. 클라이언트 사이드 리다이렉트는 UX 목적이지 보안 경계가 아닙니다. 페이지 콘텐츠는 인증 확정 전까지 숨기고, 모든 데이터 접근 거부는 서버 사이드 규칙에서 수행해야 합니다.

CISIXERS-005

Weak email-domain validation; missing Google hd parameter

HIGH | CVSS 3.1 / 7.1 | AV:N/AC:L/PR:N/UI:R/S:U/C:H/I:L/A:N | CWE-20 | CWE-287

SUMMARY

EN Domain validation is performed with `email.endsWith('@chadwickschool.org')` on the client side only, and no `hd` parameter is passed to `GoogleAuthProvider`. Two issues follow: (a) the provider accepts any Google account and merely revokes the session afterward (see CISIXERS-003); (b) string-based suffix matching is fragile — null/undefined emails throw at runtime, plus-address aliases and Unicode homoglyphs in display paths can confuse downstream logic.

EVIDENCE

```
if (!email.endsWith('@chadwickschool.org')) { ... }
// Passes for: evil+anything@chadwickschool.org
// Throws if email is null/undefined - no try/catch guard
```

IMPACT

Combined with CISIXERS-003, any Google user can authenticate. Even with the client check working as intended, silent exceptions can cause the validation branch to be skipped entirely.

REMEDIATION

```
// Provider-level domain restriction (Google Workspace enforces this)
googleProvider.setCustomParameters({ hd: 'chadwickschool.org' });

// Defensive client check as secondary validation
const EMAIL_REGEX = /^[a-z0-9._%+-]+@chadwickschool\.org$/;
function isValidSchoolEmail(email) {
  return typeof email === 'string' && EMAIL_REGEX.test(email.toLowerCase());
}
```

Primary enforcement must still occur in Firebase Security Rules against `request.auth.token.email` so that even bypassed client logic cannot read data.

KR 도메인 검증이 `endsWith` 로만 이루어지고 `GoogleAuthProvider` 에 `hd` 파라미터가 설정되어 있지 않습니다. 이로 인해 ① 모든 Google 계정이 OAuth 플로우를 통과한 뒤 사후적으로만 제거되고, ② null·undefined 이메일에서 런타임 예외가 발생하거나 plus-alias 등 엣지 케이스가 우회 가능성을 만듭니다. 제공자 차원의 `hd` 제한과 서버 사이드 규칙 검증을 결합해야 완전한 방어가 됩니다.

CISIXERS-006

Firebase Storage rules state unverified

HIGH | CVSS 3.1 / 6.8* (conditional) | CWE-284

SUMMARY

EN The Firebase config references `storageBucket: "[firebase-storage-redacted]"`, but no `storage.rules` file is version-controlled. In Firebase's test-mode default, any authenticated user can read and write any object in the bucket, including overwriting other users' uploads.

IMPACT

- Arbitrary file upload and overwrite across the bucket (users overwriting each other's avatars, images, attachments).
- Potential hosting of malicious files under the school's domain.
- Enumeration of all existing file paths if list is allowed.

REMEDIATION

```
rules_version = '2';
service firebase.storage {
  match /b/{bucket}/o {
    function isChadwick() {
      return request.auth != null
        && request.auth.token.email.matches('.*@chadwickschool[.]org$');
    }

    match /avatars/{uid}/{fileName} {
      allow read: if isChadwick();
      allow write: if isChadwick() && request.auth.uid == uid
        && request.resource.size < 5 * 1024 * 1024
        && request.resource.contentType.matches('image/.*');
    }

    match /{allPaths=**} {
      allow read, write: if false;
    }
  }
}
```

> KR Firebase 구성에 `storageBucket`이 존재하나, `storage.rules`가 버전 관리되고 있지 않습니다. 테스트 모드 기본값이 유지되고 있다면 인증된 사용자라면 누구나 버킷의 모든 객체를 읽고 덮어쓸 수 있으며, 타인의 업로드를 덮어쓰거나 학교 도메인 하에 악성 파일을 호스팅할 위험이 존재합니다. 경로별 소유자 스코프와 MIME·크기 제한을 포함한 명시적 규칙으로 전환해야 합니다.

CISIXERS-007

Sensitive user data cached in localStorage (XSS amplifier)

MEDIUM | CVSS 3.1 / 6.1 | AV:N/AC:L/PR:N/UI:R/S:C/C:L/I:L/A:N | CWE-922 | CWE-359

SUMMARY

EN `auth.js` duplicates the authenticated user's `uid`, display name, email, and photo URL into `localStorage`. Firebase Auth already persists session state securely via IndexedDB; the additional copy provides no benefit and increases blast radius for any future XSS finding.

IMPACT

- Any script execution on any page of the site (XSS, third-party CDN compromise, malicious browser extension) can exfiltrate `uid` and identity data with one `localStorage.getItem('user')` call.
- Values are writable by attacker scripts, enabling identity spoofing for client-side logic that trusts `localStorage`.

REMEDIATION

Remove `localStorage.setItem('user', ...)` entirely. Read the current user via `auth.currentUser` or subscribe to `onAuthStateChanged`.

> KR `auth.js` 는 인증된 사용자의 식별 정보(uid, 이름, 이메일, 사진)를 `localStorage` 에 중복 저장합니다. Firebase Auth는 IndexedDB를 통해 세션을 자체적으로 안전하게 보관하므로 추가 복사는 불필요하며, XSS·CDN 침해·악성 브라우저 확장 등 JavaScript 실행 경로가 생기는 어떤 경우에도 신원 정보가 한 줄로 유출됩니다. 해당 저장 로직을 제거하고 `auth.currentUser` 를 참조해야 합니다.

CISIXERS-008

Default SESSION_SECRET placeholder committed

MEDIUM | CVSS 3.1 / 5.9 | CWE-1188 | CWE-798

SUMMARY

EN `.env` contains `SESSION_SECRET=your_session_secret_key_here`. If any deployed process ever reads this placeholder as its signing secret, session tokens become forgeable by anyone who has seen the placeholder value — which is now the entire internet.

IMPACT

If a server (current or future) uses this secret to sign session cookies or JWTs, arbitrary session forgery is possible. The current live deployment appears frontend-only, but the presence of `NODE_ENV` and `PORT` in the same file indicates an intended or historical backend.

REMEDIATION

Generate a cryptographically random secret per environment and store it in a secret manager (GitHub Actions secrets, Google Secret Manager, etc.) rather than in the repository.

```
node -e "console.log(require('crypto').randomBytes(64).toString('hex'))"
```

> KR `.env` 의 `SESSION_SECRET` 값이 플레이스홀더 그대로입니다. 서버 프로세스가 이 값을 실제 서명 키로 사용한 적이 있거나 사용한다면, 해당 플레이스홀더를 본 모든 사람이 세션 토큰을 위조할 수 있습니다. 환경별로 충분한 엔트로피를 가진 값을 생성하여 GitHub Actions secrets 또는 Google Secret Manager 등 별도 비밀 관리 체계에 저장해야 합니다.

CISIXERS-009

Missing security headers

MEDIUM | CVSS 3.1 / 5.4 | CWE-693 | CWE-1021

SUMMARY

EN HTTP response headers lack `Content-Security-Policy`, `Strict-Transport-Security`, `X-Frame-Options`, `X-Content-Type-Options`, `Referrer-Policy`, and `Permissions-Policy`. This increases the impact of any XSS or downgrade attack and allows clickjacking via iframe embedding.

EVIDENCE

```
$ curl -I https://[redacted].org
HTTP/2 200
server: GitHub.com
content-type: text/html; charset=utf-8
access-control-allow-origin: *
# No CSP / HSTS / X-Frame-Options / X-Content-Type-Options / Referrer-Policy
```

REMEDIATION

GitHub Pages does not permit custom response headers, so use `<meta http-equiv>` directives where supported and place a CDN (Cloudflare free tier) in front to inject the remainder.

```
<meta http-equiv="Content-Security-Policy"
  content="default-src 'self';
           script-src 'self' https://www.gstatic.com https://cdnjs.cloudflare.com;
           style-src 'self' 'unsafe-inline' https://cdnjs.cloudflare.com;
           img-src 'self' data: https:;
           connect-src 'self' https://*.googleapis.com https://*.firebaseio.com;
           frame-ancestors 'none';">
<meta http-equiv="X-Content-Type-Options" content="nosniff">
<meta name="referrer" content="strict-origin-when-cross-origin">
```

> KR HTTP 응답에 CSP·HSTS·X-Frame-Options·X-Content-Type-Options·Referrer-Policy·Permissions-Policy 헤더가 전무합니다. XSS 발생 시 피해 확산, SSL 다운그레이드, iframe 클릭재킹 위험이 커집니다. GitHub Pages는 커스텀 헤더를 허용하지 않으므로 `<meta>` 지시자와 앞단 CDN(Cloudflare 무료 플랜의 Transform Rules 등)을 결합해 주입해야 합니다.

CISIXERS-010

Missing email_verified claim check

MEDIUM | CVSS 3.1 / 5.3 | CWE-287

SUMMARY

EN Authentication accepts any Google-authenticated user whose email ends with `@chadwickschool.org` without checking the `email_verified` claim. Although Google Workspace normally sets this to `true`, defense in depth dictates explicit verification.

REMEDIATION

```
if (!user.emailVerified || !isValidSchoolEmail(user.email)) {
  await signOut(auth);
  return;
}
```

Also enforce at the rule level: `request.auth.token.email_verified == true`.

> KR 현재 인증 로직은 Google 로그인 결과의 `email_verified` 클레임을 확인하지 않습니다. Google Workspace에서는 일반적으로 true지만, 심층 방어 원칙에 따라 클라이언트와 Firestore 규칙 양쪽에서 명시적으로 검증해야 합니다.

CISIXERS-011

Missing Subresource Integrity (SRI) on CDN resources

LOW | CVSS 3.1 / 4.3 | CWE-353 | CWE-494

SUMMARY

EN The Font Awesome stylesheet is loaded from `cdnjs.cloudflare.com` without an `integrity` attribute. If the CDN is compromised or a downstream cache is poisoned, attackers can inject arbitrary CSS (or, for script resources, JavaScript) site-wide.

EVIDENCE

```
<link rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
<!-- No integrity="..." nor crossorigin="anonymous" -->
```

REMEDIATION

```
<link rel="stylesheet"
      href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css"
      integrity="sha512-iecdLmaskl7CVkqk..."
      crossorigin="anonymous"
      referrerpolicy="no-referrer">
```

> KR Font Awesome을 CDN에서 `integrity` 속성 없이 로드 중입니다. CDN 침해 또는 캐시 오염 시 사이트 전체에 임의 CSS(스크립트의 경우 JavaScript) 주입이 가능합니다. SRI 해시를 추가하고 `crossorigin="anonymous"` 를 함께 지정해야 합니다.

CISIXERS-012

Raw Firebase error messages exposed to users

LOW | CVSS 3.1 / 3.7 | CWE-209

SUMMARY

EN `showError('❌ Sign-in failed: ' + error.message)` propagates Firebase error strings directly to the UI. While rendered via `textContent` (so no XSS), exposed error codes enable user enumeration and fingerprinting of backend state.

REMEDIATION

```
catch (error) {
  console.error(error);                              // full detail for devs only
  showError('Sign-in failed. Please try again.');    // generic to users
}
```

> KR Firebase 에러 메시지가 사용자 UI에 그대로 노출됩니다. `textContent` 로 렌더링되어 XSS는 차단되나, 에러 코드 공개로 인해 계정 열거·백엔드 상태 추정이 가능합니다. 사용자에게는 일반 메시지, 개발자에게는 console 로그로 분리하십시오.

CISIXERS-013

.DS_Store committed to repository

LOW | CVSS 3.1 / 3.1 | CWE-540 | CWE-200

SUMMARY

EN A macOS `.DS_Store` file is committed. It encodes local filesystem metadata and can reveal names of files that were present during development but later removed or renamed (e.g., leftover `secret.txt` filenames, draft documents).

REMEDIATION

```
git rm --cached .DS_Store
echo ".DS_Store" >> .gitignore
git commit -m "Remove .DS_Store and ignore"
```

Also purge from history alongside CISIXERS-001 remediation.

> KR macOS의 `.DS_Store` 가 커밋되어 있으며, 이 파일은 개발 당시 존재했던 파일명(예: 임시 secret 파일, 초고 문서) 등을 역추적 가능한 형태로 노출합니다. 캐시에서 삭제 후 `.gitignore` 에 추가하고, CISIXERS-001의 히스토리 삭제 작업과 함께 영구 제거해야 합니다.

CISIXERS-014

OAuth callback URL uses insecure http://

LOW | CVSS 3.1 / 3.1 | CWE-319

SUMMARY

EN `CALLBACK_URL=http://[redacted].org/auth/google/callback` specifies an unencrypted scheme. While the current frontend does not exercise this callback (Firebase popup flow is in use), any future enablement of the server-side flow would transmit OAuth authorization codes in plaintext over an attacker-observable channel.

REMEDIATION

Use `https://` exclusively. Configure this in the Google Cloud OAuth consent configuration and in all `.env` variants.

> KR OAuth 콜백 URL이 평문 `http://` 스킴으로 지정되어 있습니다. 현재는 Firebase 팝업 플로우를 사용하므로 실제로 호출되지 않지만, 서버 사이드 플로우로 전환할 경우 OAuth authorization code가 평문으로 전송되어 중간자 노출 대상이 됩니다. Google Cloud OAuth 콘솔과 환경 변수 양쪽에서 `https://` 로 통일해야 합니다.

CISIXERS-015

Repository URL publicly disclosed via school announcement

INFORMATIONAL | CWE-200

SUMMARY

EN The source repository URL was shared in a school-wide announcement. This is not a vulnerability in itself — open source is valid — but it means every observation in this report is trivially reachable by any student. Combined with the findings above, the effective attack surface is broader than a typical obscure-URL project.

RECOMMENDATION

Continue with a public repository after remediation, but: treat every commit as if adversaries will read it in minutes; enable GitHub secret scanning alerts; consider adding a `SECURITY.md` with a private disclosure contact; remove the `.env` (CISIXERS-001) before the repository is next advertised.

> KR 소스 리포지토리 주소가 학교 전체 공지를 통해 공개되었습니다. 오픈 소스 자체가 문제는 아니지만, 본 보고서의 모든 관찰 사항이 모든 재학생에게 즉시 도달 가능한 상태라는 의미이며, 앞선 취약점들과 결합될 때 실질 공격 표면이 일반적인 프로젝트보다 넓어집니다. 리포지토리 공개를 유지하되 ① 모든 커밋이 즉시 외부에 읽힘을 전제로 삼고, ② GitHub Secret Scanning Alerts를 활성화하며, ③ 비공개 제보 채널을 명시한 `SECURITY.md` 를 추가하고, ④ 다음 공개 전에 CISIXERS-001의 `.env` 제거를 완료하는 것을 권장합니다.

5. Remediation Priorities

EN The following sequence minimizes exposure window and correctly orders dependencies between fixes.

Within 1 hour

1. Reset the Google OAuth Client Secret in Google Cloud Console (CISIXERS-001).
2. Verify Firestore and Storage rules in Firebase Console; if in test mode, switch to a deny-by-default policy immediately (CISIXERS-002, CISIXERS-006).

Within 24 hours

1. Purge `.env` and `.DS_Store` from git history; force-push (CISIXERS-001, CISIXERS-013).
2. Add `hd: 'chadwickschool.org'` to the Google provider (CISIXERS-003, CISIXERS-005).
3. Replace the "sign in then sign out" pattern with pre-auth-gated rendering and server-side rule enforcement (CISIXERS-003, CISIXERS-004).
4. Audit Google Cloud audit logs and Firestore usage logs for suspicious activity during the exposure window.

Within 1 week

1. Remove `localStorage` duplication of user identity (CISIXERS-007).
2. Add security headers via `<meta>` and consider a Cloudflare proxy for HSTS/HTTP-only headers (CISIXERS-009).
3. Add SRI attributes to CDN-loaded resources (CISIXERS-011).
4. Sanitize error messages (CISIXERS-012).
5. Version-control `firestore.rules`, `storage.rules`, and `firebase.json` and deploy via `firebase deploy`.

> 조치 우선순위 요약
>
> **1**시간 이내**:** Google OAuth Client Secret 재발급, Firestore·Storage 규칙 점검 및 필요 시 deny-by-default 즉시 전환.
>
> **24**시간 이내**:** `.env` · `.DS_Store` 히스토리 영구 삭제, Google `hd` 파라미터 적용, 인증 게이팅 구조 리팩터링, Google Cloud·Firestore 감사 로그 점검.
>
> **1**주 이내**:** localStorage 사용자 중복 저장 제거, 보안 헤더 주입, SRI 추가, 오류 메시지 일반화, Firebase 규칙 버전 관리 체계 도입.

6. Timeline & Disclosure

DATE (KST)	EVENT
2026-04-18 13:20	GitHub Pages rate-limit placeholder page observed; initial public-surface reconnaissance began.
2026-04-18 13:30	HTTP headers, DNS, and static HTML analysis completed.
2026-04-18 13:45	`auth.js` reviewed; client-side authentication model identified.
2026-04-18 14:00	Repository URL obtained (previously disclosed in public school announcement).
2026-04-18 14:05	`.env` discovered in repository root — Client Secret exposure confirmed.
2026-04-18 14:10	Report compilation begun. Repository owner to be notified concurrently.

Disclosure posture: This report was delivered privately to the repository owner first, with at least 72 hours allowed for the Critical findings to be remediated before any broader sharing. The researcher has no intention of publishing exploits,

PoCs, or the leaked secret value outside this document. This public release version has additionally redacted target identifiers, repository owner, and project IDs.

> 공개 방침**:** 본 보고서는 리포지토리 소유자에게 우선 비공개로 전달되었으며, Critical 등급 항목에 대한 조치를 위해 최소 72시간의 여유를 둔 뒤에만 보다 넓은 범위로 공유되었습니다. 연구자는 본 문서 외부에 익스플로잇, PoC, 또는 노출된 비밀 값을 게시할 의도가 없으며, 본 공개 버전에서는 추가적으로 대상 식별자, 리포지토리 소유자, 프로젝트 ID를 redact 처리하였습니다.

7. Appendix

A. Researcher

Name	Geonhee (건희)
Role	Independent security researcher (student)
Contact	(provided to repository owner out-of-band)

B. Tools Used

- `curl`, `dig` — HTTP and DNS reconnaissance
- Browser DevTools — source inspection, runtime analysis
- GitHub web interface — repository and history review

C. Acknowledgement of Consent

EN All activity described in this report was conducted on assets whose source repository was voluntarily and publicly disclosed by the owner via a school-wide announcement, and was non-invasive passive review of public material. No exploitation against live user data was performed, no credentials were tested, and no user sessions were compromised in the preparation of this document.

> KR 본 보고서에 기술된 모든 행위는 리포지토리 소유자가 학교 전체 공지를 통해 소스 주소를 자발적으로 공개한 자산을 대상으로, 공개된 자료에 대한 비침투적 수동 리뷰로 수행되었습니다. 실사용자 데이터에 대한 공격 실행, 자격증명 시도, 사용자 세션 탈취 등은 본 문서 작성 과정에서 일절 수행되지 않았습니다.

D. Glossary

TERM	MEANING
CVSS	Common Vulnerability Scoring System (v3.1) — industry-standard severity scale.
CWE	Common Weakness Enumeration — classification of software weakness types.
BaaS	Backend-as-a-Service — e.g. Firebase, Supabase.
RLS	Row Level Security — database-side access-control policies.
SRI	Subresource Integrity — browser feature to verify CDN-loaded resources by hash.
hd (OAuth)	Google OAuth parameter restricting sign-in to a specific hosted-domain (G Suite/Workspace).

E. Redaction Key

The following placeholders replace sensitive identifiers in this public-release version. The original report delivered to the repository owner contains the actual values.

PLACEHOLDER	WHAT IT STOOD FOR
`[redacted].org`	Target site domain
`[peer]`	Repository owner's GitHub username
`[redacted]` (repo)	Repository name
`[upstream-author]/[upstream-repo]`	Upstream fork parent
`[firebase-project-redacted]`	Firebase project ID
`[firebase-storage-redacted]`	Firebase Storage bucket
`[gcp-project-redacted]`	Google Cloud project number
`[client-id-redacted]`	OAuth 2.0 Client ID
`[secret-rotated]`	OAuth Client Secret (rotated post-disclosure)

— End of Report — 보고서 끝 —

Generated April 18, 2026 · Redacted version released for portfolio publication