Secure CTF Platform

Cybersecurity Project - Next.js + Firebase

Romain Herrenknecht, Stanislas Thibaud, Gaspard Auclair,
Mael Tellus, Romain Oualid

ISEP - Cybersecurity Course
January 2026

Problem and Goals

What the platform enables

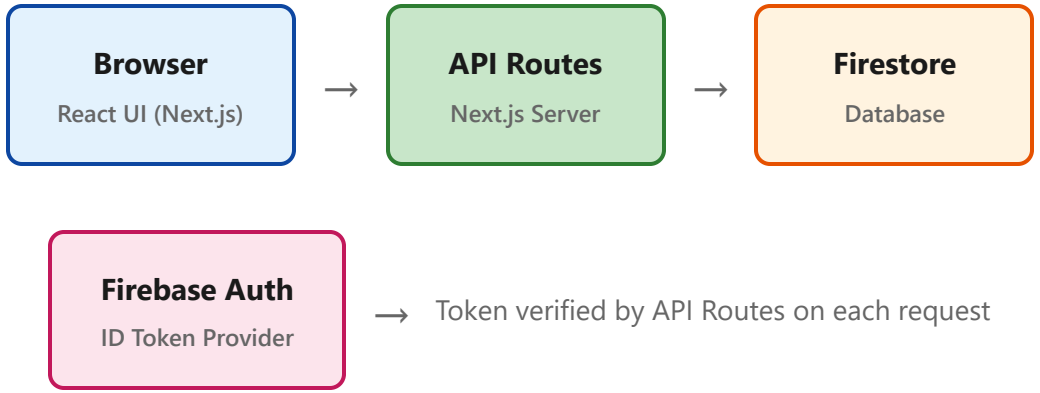
- Multi-user Capture The Flag competition platform
- Users register, solve challenges, and submit flags
- Automatic scoring based on challenge difficulty
- Real-time leaderboard and public player profiles

Security Goal

> **Core Principle:** No critical logic on the client. All sensitive operations (flag verification, score updates, authentication) are executed server-side only.

- Never trust client input
- Flags are never stored or transmitted in plaintext
- Defense-in-depth with multiple security layers

Architecture Overview



Key Principles

- **Trust boundary:** Client is untrusted; server validates everything
- **Server-side validation:** Firebase Admin SDK verifies ID tokens
- **Defense-in-depth:** Firestore Rules + API validation + rate limiting

Authentication and Access Control

Providers

- Email / Password
- Google OAuth
- GitHub OAuth

Password Policy

- Minimum 12 characters
- At least 1 uppercase letter
- At least 1 lowercase letter
- At least 1 digit
- At least 1 special symbol

Access Control

- **Email verification required** before accessing `/challenges` and `/leaderboard`
- Protected routes redirect to verification page

Anti-Enumeration

> Generic error messages prevent attackers from discovering valid emails or usernames.

Data Model (Firestore)

Collection	Key Fields
`users/{uid}`	displayName, score, solvedChallenges[], cheatedChallenges[], emailVerified, isBanned
`challenges/{id}`	title, category, difficulty, points, **flagHash**, hints[], isComingSoon
`submissions/{id}`	userId, challengeId, status, **ipHash**, timestamp

Security Considerations

- **No plaintext flags:** Only salted SHA-256 hashes stored in `flagHash`
- **Minimal PII:** Public profiles expose only displayName and score (no emails)
- **IP hashing:** IP addresses are hashed before logging for privacy

Flag Verification (Server-Side)

Verification Flow

1. Receive flag from authenticated user
2. Normalize: `flag.trim().toLowerCase()`
3. Hash: `SHA256(SALT + flag)`
4. Compare with `flagHash` (timing-safe)
5. If correct: transaction update score
6. Log submission (ipHash, timestamp)
7. Return generic response

Security Guarantees

- **No client-side verification**
- **Score computed server-side**
- **Timing-safe comparison**
- **Atomic transaction** (no double scoring)

> Flag is NEVER sent back to client. Only a boolean result.

Rate Limiting and Abuse Prevention

Configuration

Endpoint	Limit
Flag submission	5 / minute
Authentication	10 / minute
General API	100 / minute

Implementation

- Sliding window algorithm
- Key: `ipHash:userId`
- Prevents bypass via IP change or multi-account

Response Strategy

- HTTP 429 + `Retry-After` header
- Consistent response time (no timing leaks)

Production Note

> In-memory Map sufficient for single-instance demo.
> Production: **Redis** for distributed limiting.

Firestore Rules / Defense in Depth

Principle

Client SDK access is restricted. All writes use Admin SDK via API Routes.

Access Matrix

Collection	Client Read	Client Write
users	Auth only	Denied
challenges	Verified email	Denied
submissions	Own only	Denied

flagHash Protection

Challenge data fetched via API Routes only. API filters `flagHash` before response. Client SDK never queries challenges directly.

Rules Snippet

```
match /users/{userId} {
  allow read: if request.auth != null;
  allow write: if false;
}
match /challenges/{id} {
  allow read: if request.auth.token.email_verified;
  allow write: if false;
}
match /{doc=**} {
  allow read, write: if false;
}
```

Deployment and Operational Security

Infrastructure

- **Vercel:** Automatic HTTPS, global CDN
- **Firebase Auth:** Authorized domains configured
- **Environment variables:** Admin SDK credentials stored server-side only

Secrets Management

- `FIREBASE_ADMIN_*` keys in Vercel env
- `FLAG_HASH_SALT` for flag hashing
- Never exposed to client bundle

Security Headers

`Strict-Transport-Security`	HSTS
`X-Frame-Options`	DENY
`X-Content-Type-Options`	nosniff
`Referrer-Policy`	strict-origin
`Content-Security-Policy`	Restrictive CSP

Demo and Conclusion

Demo Steps

1. Register a new account
2. Verify email address
3. Browse challenges (filter by category)
4. Solve a challenge (e.g., Base64 Basics)
5. Observe score update
6. Check leaderboard ranking
7. View public profile

Conclusion

- **Architecture separation:** Client handles UI only; server handles all critical logic
- **Server-side verification:** Flags, scores, and auth are never trusted from client
- **Objectives met:** Multi-user CTF with secure auth, scoring, and ranking

Future Improvements

- Admin panel for challenge management
- Redis-based distributed rate limiting
- Real-time leaderboard with WebSockets