

continue fixing from bug 5

## Comprehensive Bug Report: PROTECT THE STREAMS

### CRITICAL BUGS (App-Breaking)

BUG 1: useMintStreamer.ts - require('bs58') crashes in client-side code

File: useMintStreamer.ts:206-207

```
function userFriendlySignature(sig: Uint8Array): string {  
  const bs58 = require('bs58');  
  return bs58.encode(sig);  
}
```

Problem: require() is used in a "use client" module. In Next.js client bundles, require of a CommonJS module like bs58 can fail or behave unpredictably. The bs58 v6 package (you have bs58@6.0.0) exports only ESM and does not support require() at all. This will throw ERR\_REQUIRE\_ESM at runtime, crashing every successful mint when trying to display the signature.

Fix: Import bs58 at the top of the file with import bs58 from 'bs58' and use it directly in the helper.

BUG 2: usePvPMatchmaking.ts - Infinite re-subscribe loop (dependency on myStats object)

File: usePvPMatchmaking.ts:149

```
}, [enabled, streamerId, sessionId, playerId, wager, myStats, playerName]);
```

Problem: myStats is a new object reference on every render (created via applyNatureToStats or baseStreamer.stats spread on line 27-29). Since it's in the useEffect dependency array, the effect tears down and re-subscribes the Supabase channel on every render. This causes:

Constant channel subscribe/unsubscribe flickering

Matchmaking never stabilizes long enough to find an opponent

Potential Supabase rate-limiting

Memory leaks from orphaned channel subscriptions

Fix: Memoize myStats with useMemo, or serialize it into the dependency array (e.g., JSON.stringify(myStats)).

BUG 3: usePvPBattle.ts - handleAction in useEffect dependency causes infinite re-subscription

File: usePvPBattle.ts:459

```
}, [matchId, opponentId, playerId, myStreamer, handleAction, battleStatus,
turnLocked, sendAction, isSpectator]);
```

Problem: `handleAction` is a `useCallback` that depends on `opponent`, `player.hp`, `isTurn` — all of which change frequently during battle. Every time these change, `handleAction` gets a new reference, causing the entire Supabase channel subscription `useEffect` to re-run, tearing down and re-creating the battle channel mid-fight. This causes:

Lost actions/moves during battle

Desynchronization between players

"SYNC\_ERROR" messages

Potential duplicate move processing

Fix: Use refs for the values that `handleAction` needs (you already have `playerRef`, `opponentRef` etc.), and remove `handleAction` from the dependency array by stabilizing it, or move the subscription logic to use a ref-based callback.

BUG 4: `usePvPBattle.ts` - Stale `isComplete` in bot retaliation timeout

File: `usePvPBattle.ts:499-525`

```
setTimeout(() => {
  if (isComplete) return; // <-- captures closure value, always false
  ...
}, 2000);
```

Problem: The `isComplete` check inside the `setTimeout` captures the value from the closure at call time (always `false` since you just set it). If the player kills the bot on their turn (setting `isComplete` = `true` via `setIsComplete`), the bot retaliation still fires 2 seconds later, potentially reducing player HP below 0 and setting a second winner.

Fix: Use a ref for `isComplete` or check the state inside the callback via a ref.

BUG 5: `useCollectionStore.ts` - `syncFromCloud` ignores falsy values (0, empty arrays, `false`)

File: `useCollectionStore.ts:331-351`

```
syncFromCloud: (userData: any) => {
  if (userData.xp) updates.totalResistanceScore = userData.xp;
  if (userData.level) updates.level = userData.level;
  if (userData.wins) updates.wins = userData.wins;
  if (userData.losses) updates.losses = userData.losses;
```

Problem: Truthy checks (`if (userData.xp)`) fail when the value is 0. A new user with

xp: 0, level: 0, wins: 0, losses: 0 will have those values silently skipped, preserving stale localStorage data instead of syncing the real cloud state. This means:

A user who lost all PvP matches and has wins: 0 will still show their old cached win count

New account with 0 XP won't overwrite cached XP from a previous session

Fix: Use !== undefined checks instead of truthy checks.

## HIGH SEVERITY BUGS

BUG 6: usePvPBattle.ts - Writing stale player.hp to database during init

File: usePvPBattle.ts:363-366

```
await supabase.from('pvp_matches').update({
  [hpCol]: player.hp, // <-- this is the INITIAL hp, not the recovered hp
}).eq('id', matchId);
```

Problem: This writes player.hp from the state at mount time, but the HP was just updated from the database a few lines above via setPlayer. Because React state updates are async, player.hp still holds the initial value (calculated max HP), not the recovered HP from the database. This overwrites the saved battle HP with full HP on every reconnect, essentially giving free heals on page refresh.

Fix: Use the mySavedHp value from the database fetch instead of player.hp.

BUG 7: player/sync/route.ts - PvP wins/losses overwrite server values with client values

File: player/sync/route.ts:100-101

```
if (wins !== undefined) updates.wins = wins;
if (losses !== undefined) updates.losses = losses;
```

Problem: The client sends absolute wins/losses counts, and the server blindly accepts them. A player could send { wins: 9999 } and it would be written. This is a server-authoritative bypass — the server should increment, not accept absolute values from the client.

Fix: Use server-side increment: wins: (user.wins || 0) + 1 when a win is reported, or better, only update wins/losses via the server-side PvP RPC.

BUG 8: player/sync/route.ts - securedIds can be injected by client

File: player/sync/route.ts:102

```
if (securedIds !== undefined) updates.secured_ids = securedIds;
```

Problem: A client can send any array of streamer IDs as securedIds without the server verifying that the user actually minted those NFTs. A user could claim to own all 20 streamers by posting { securedIds: ["kaicenat", "speed", ...] }.

Fix: Verify on-chain NFT ownership before accepting securedIds, or don't allow the client to set this directly.

BUG 9: useResistanceMission.ts - Player/enemy state initialized with useState won't update on prop change

File: useResistanceMission.ts:84-99

```
const [player, setPlayer] = useState<EntityState>({
  ...
  maxHp: calculatedMaxHp,
  hp: calculatedMaxHp,
  stats: modifiedStats,
});

const [enemy, setEnemy] = useState<EntityState & { moves?: any[] }>({
  ...
  maxHp: 120 + (threatLevel * 20),
  hp: 120 + (threatLevel * 20),
});
```

Problem: useState initializers only run on first mount. If the user navigates away and opens a new mission with a different streamer (or if threatLevel changes), the player/enemy state is stale from the previous battle. The component doesn't reset when streamer changes because React reuses the same component instance.

Fix: Add a key={streamer.id} to <MissionTerminal> in the parent, or use a useEffect to reset state when streamer.id changes.

BUG 10: useMintStreamer.ts - Fallback transfer takes user's SOL but doesn't actually mint an NFT

File: useMintStreamer.ts:153-179

```
// FALLBACK TO CLIENT-SIDE TRANSFER IF API FAILS
const transaction = new Transaction().add(
  SystemProgram.transfer({
    fromPubkey: publicKey!,
    toPubkey: new PublicKey(CONFIG.TREASURY_WALLET),
    lamports: CONFIG.MINT_PRICE * LAMPORTS_PER_SOL,
  })
)
```

```
);
```

```
...
```

```
secureAsset(streamerId); // Marks as "owned" locally
```

Problem: The fallback sends SOL to the treasury and marks the asset as "secured" locally, but no NFT is ever minted on-chain. The user pays but gets nothing verifiable. Additionally, secureAsset syncs this to the cloud as if they own it, but there's no on-chain proof. If the user checks their wallet later, the NFT doesn't exist.

Fix: Either remove the fallback (safest), or queue a server-side retry to mint the NFT later, or at minimum warn the user clearly that this is a payment without an NFT.

BUG 11: auth.ts - Hardcoded JWT secret fallback

File: auth.ts:3

```
const SECRET_KEY = new TextEncoder().encode(process.env.JWT_SECRET ||  
'pts-resistance-secret-key-2026');
```

Problem: If JWT\_SECRET is not set in production, the fallback 'pts-resistance-secret-key-2026' is publicly visible in the source code. Anyone can forge JWT tokens and impersonate any user.

Fix: Throw an error at startup if JWT\_SECRET is not set in production.

## MEDIUM SEVERITY BUGS

BUG 12: usePvPMatchmaking.ts - retry() doesn't actually restart matchmaking

File: usePvPMatchmaking.ts:151-153

```
const retry = () => {  
  setMatchStatus('SEARCHING');  
};
```

Problem: retry only sets the status to SEARCHING but doesn't re-subscribe to the Supabase channel or re-track presence. The useEffect that handles matchmaking depends on enabled, not matchStatus, so changing matchStatus alone does nothing. The user clicks retry but nothing happens.

Fix: Either toggle enabled off and on, or use a counter/key to force the effect to re-run.

BUG 13: useResistanceMission.ts - executeMove uses stale enemy in addLog

File: useResistanceMission.ts:385

```
addLog(`Inflicted ${damage} disruption to ${enemy.name}.`);
```

Problem: This log is added after `setEnemy(prev => ...)` potentially transitions the enemy. The `enemy.name` in the closure is from before the state update, which is usually fine, but during stage transitions (enemy defeated → new enemy spawned), it may log the new enemy's name since the `setEnemy` call in the timeout hasn't resolved yet. More importantly, the enemy dependency in the `useCallback` is the full enemy object, causing the callback to recreate on every enemy HP change.

BUG 14: `useResistanceMission.ts` - `handleEnemyTurn` triggers `markMissionComplete` on death with no guard against double-calling  
File: `useResistanceMission.ts:208-213`

```
if (newHp === 0) {  
  setIsComplete(true);  
  setResult('FAILURE');  
  markMissionComplete(streamer.id, 'F', 10);  
}
```

Problem: Unlike the SUCCESS path (which has `hasMarkedComplete` guard), the FAILURE path calls `markMissionComplete` directly. If React batches the `setIsComplete(true)` update and the enemy turn timer fires again before the state updates, `markMissionComplete` could be called twice for the same failed mission.

BUG 15: `page.tsx` - `mint()` called without streamer ID from hero button  
File: `page.tsx:220`

```
onClick={() => { playClick(); mint(); }}
```

Problem: The hero "INITIALIZE MINT" button calls `mint()` with no `streamerId` argument. Looking at `useMintStreamer`, this hits the early return at line 31-35:

```
if (!streamerId) {  
  setError({ code: "NO_TARGET", message: "No streamer target selected." });  
  return;  
}
```

The button appears to do something but always errors. Users who don't scroll to the roster will be confused.

Fix: Either remove the hero mint button, or make it mint a random streamer, or make it scroll to the roster.

BUG 16: useGameDataStore.ts - Supabase fetch fails silently and breaks game data

File: useGameDataStore.ts:28-32

```
const { data: streamersData, error: streamersError } = await supabase
  .from('streamers').select('*');
```

```
if (streamersError) throw streamersError;
```

Problem: If the streamers, moves, or items tables don't exist in Supabase (or env vars are placeholder values), this throws and the catch block sets `isInitialized: true` but keeps the local fallback data. However, if Supabase returns empty data (tables exist but are empty), the code processes empty arrays and sets `processedStreamers` to an empty array. The fallback check `processedStreamers.length > 0 ? processedStreamers : localStreamers` catches this, but the items fallback has the same pattern and would produce an empty items map, breaking the shop/inventory.

BUG 17: player/sync/route.ts - No rate limiting on sync endpoint

File: player/sync/route.ts:11

Problem: There's no rate limiting. A malicious user could spam `POST /api/player/sync` with `{ deltaXp: 4999 }` (just under the 5000 cap) hundreds of times per second to farm unlimited XP and PTS. The anti-cheat only checks per-request XP cap, not cumulative rate.

BUG 18: useResistanceMission.ts - Defense boost multiplier of 1.5 INCREASES damage taken

File: useResistanceMission.ts:200-203

```
if (defenseBoost.turnsLeft > 0) {
  enemyDamage = Math.floor(enemyDamage * defenseBoost.multiplier);
}
```

But the `DEFENSE_MATRIX` item sets value: 1.5:

```
DEFENSE_MATRIX: { effect: 'boostDefense', value: 1.5 }
```

And the boost is applied as:

```
setDefenseBoost({ multiplier: item.value, turnsLeft: 3 });
```

Problem: A "defense boost" with multiplier 1.5 multiplies enemy damage by 1.5, making the player take 50% MORE damage instead of less. The Defense Matrix

item actively hurts the player.

Fix: The multiplier should be 0.5 (halving damage) or the formula should be  $\text{enemyDamage} / \text{defenseBoost.multiplier}$ .

#### LOW SEVERITY BUGS

BUG 19: usePvPBattle.ts - Opponent maxHp hardcoded to 100

File: usePvPBattle.ts:343-348

```
setOpponent({
```

```
  ...
```

```
  maxHp: 100,
```

```
  hp: oppHp,
```

Problem: Opponent's maxHp is always 100 regardless of their actual level/stats. HP bars will show incorrect percentages if the opponent has leveled up and has more than 100 max HP.

BUG 20: page.tsx - onSectorClick casts sector data as Streamer without validation

File: page.tsx:293

```
<ResistanceMap onSectorClick={(s) => setActiveMissionStreamer(s as  
Streamer)} />
```

Problem: The sector data from ResistanceMap is cast to Streamer with as Streamer. If the map returns a sector object that doesn't match the full Streamer interface (missing moves, ultimateMove, stats, etc.), the MissionTerminal will crash when trying to access streamer.moves.forEach(...).

BUG 21: supabaseClient.ts - Placeholder URL and key allow silent failures

File: supabaseClient.ts:3-4

```
const supabaseUrl = process.env.NEXT_PUBLIC_SUPABASE_URL || 'https://  
placeholder.supabase.co';
```

```
const supabaseAnonKey = process.env.NEXT_PUBLIC_SUPABASE_ANON_KEY ||  
'placeholder-key';
```

Problem: If env vars aren't set, Supabase client is created with placeholder values. All Supabase operations silently fail with network errors rather than giving a clear "not configured" message.

BUG 22: useCheckUplinkStatus.ts - Double scan on mount

File: useCheckUplinkStatus.ts:97-106



```
useEffect(() => { scanWallet(); }, [scanWallet]);
useEffect(() => {
  if (connected && walletPubkey) { scanWallet(); }
}, [connected, walletPubkey, scanWallet]);
```

Problem: Two effects both call scanWallet() on mount when the wallet is connected, causing a redundant RPC call. This wastes rate-limited Solana RPC quota.

BUG 23: useUserAuth.ts - setUserId not in useCallback deps for login  
File: useUserAuth.ts:107

```
}, [publicKey, signMessage, syncFromCloud, setAuthenticated, setUserId]);
```

Problem: setUserId is a React state setter (from useState) and is stable by identity, so this won't cause bugs, but setUserId shouldn't be in the deps array — it's unnecessary and adds confusion.

BUG 24: SolanaProvider.tsx - Network type mismatch  
File: SolanaProvider.tsx:17

```
const network = (process.env.NEXT_PUBLIC_SOLANA_NETWORK as
WalletAdapterNetwork) || WalletAdapterNetwork.Devnet;
```

Problem: CONFIG.NETWORK uses 'devnet' or 'mainnet-beta', but WalletAdapterNetwork enum uses 'devnet' / 'mainnet-beta'. If someone sets NEXT\_PUBLIC\_SOLANA\_NETWORK=mainnet-beta, this works, but the mint API route (route.ts:12-14) uses CONFIG.NETWORK which reads from a different env var. These two could diverge (wallet on devnet, mint API on mainnet or vice versa), causing transaction failures.

BUG 25: mint/transaction/route.ts - Module-level Umi initialization crashes the API if env var is missing  
File: route.ts:19-26

```
const BACKEND_PRIVATE_KEY =
JSON.parse(process.env.BACKEND_WALLET_PRIVATE_KEY || '[]');
const keypair = umi.eddsa.createKeypairFromSecretKey(new
Uint8Array(BACKEND_PRIVATE_KEY));
const backendSigner = createSignerFromKeypair(umi, keypair);
```

Problem: If BACKEND\_WALLET\_PRIVATE\_KEY is not set, this creates a keypair from an empty array, which will throw. Since this runs at module load time (not

inside the handler), it crashes the entire API route module, making ALL requests to /api/mint/transaction return 500.

## Summary Table

#    Severity   File   Bug

- 1    CRITICAL useMintStreamer.ts require('bs58') crashes ESM-only package in client
  - 2    CRITICAL usePvPMatchmaking.ts    Infinite re-subscribe loop from myStats object reference
  - 3    CRITICAL usePvPBattle.ts        Channel re-created every state change via handleAction dep
  - 4    CRITICAL usePvPBattle.ts        Stale isComplete in bot retaliation timeout
  - 5    CRITICAL useCollectionStore.ts    syncFromCloud ignores 0/falsy values
  - 6    HIGH       usePvPBattle.ts        Writes stale HP to DB on reconnect (free heals)
  - 7    HIGH       player/sync/route.tsClient can set arbitrary wins/losses
  - 8    HIGH       player/sync/route.tsClient can inject securedIds without verification
  - 9    HIGH       useResistanceMission.tsPlayer/enemy state doesn't reset on streamer change
  - 10   HIGH       useMintStreamer.ts Fallback takes SOL but doesn't mint NFT
  - 11   HIGH       auth.ts        Hardcoded JWT secret fallback in source code
  - 12   MEDIUM   usePvPMatchmaking.ts    retry() doesn't actually restart matchmaking
  - 13   MEDIUM   useResistanceMission.tsexecuteMove callback recreates on every HP change
  - 14   MEDIUM   useResistanceMission.tsDouble markMissionComplete possible on failure
  - 15   MEDIUM   page.tsx    Hero mint button has no streamer ID (always errors)
  - 16   MEDIUM   useGameDataStore.ts    Empty Supabase tables can break items map
  - 17   MEDIUM   player/sync/route.tsNo rate limiting allows XP farming
  - 18   MEDIUM   useResistanceMission.tsDefense boost multiplier increases damage taken
  - 19   LOW       usePvPBattle.ts        Opponent maxHp hardcoded to 100
  - 20   LOW       page.tsx    Sector data cast to Streamer without validation
  - 21   LOW       supabaseClient.ts    Placeholder values cause silent failures
  - 22   LOW       useCheckUplinkStatus.ts    Double wallet scan on mount
  - 23   LOW       useUserAuth.ts        Unnecessary dep in useCallback
  - 24   LOW       SolanaProvider.tsx    Network env var divergence risk
  - 25   LOW       mint/transaction/route.tsModule-level crash if backend key missing
- The top priorities to fix are bugs 1, 2, 3, 5, and 18 — these are actively breaking minting, PvP matchmaking, PvP battles, cloud sync, and making defense items harmful. Want me to fix them?

Update Todos

Analyze modified files (mint route, page.tsx, MediaUplink, useMintStreamer, usePvPMatchmaking)

Audit API routes for bugs (auth, sync, narrate, metadata)

Audit hooks for logic errors and race conditions

Audit components for rendering/state bugs

Check config, data files, and lib utilities

Compile comprehensive bug report

fix all bugs please

I'll fix all 25 bugs systematically. Let me start with the critical ones and work through everything.

Update Todos

Fix BUG 1: require('bs58') crash in useMintStreamer.ts

Fix BUG 2: Infinite re-subscribe loop in usePvPMatchmaking.ts

Fix BUG 3: Channel re-creation loop in usePvPBattle.ts

Fix BUG 4: Stale isComplete in bot retaliation

Fix BUG 5: syncFromCloud ignores falsy values

Fix BUG 6: Stale player.hp written to DB on reconnect

Fix BUGs 7-8: Client can inject wins/losses/securedIds

Fix BUG 9: Mission state doesn't reset on streamer change

Fix BUG 10: Fallback takes SOL without minting NFT

Fix BUG 11: Hardcoded JWT secret fallback

Fix BUGs 12-18: Medium severity bugs

Fix BUGs 19-25: Low severity bugs

Edit useMintStreamer.ts

Edit failed

Read useMintStreamer.ts

Edit useMintStreamer.ts

Added 1 line

The hint is because I haven't removed the old require('bs58') yet. Let me fix that now and continue with all the other bugs.

Edit useMintStreamer.ts

Removed 1 line

Now fix BUG 10 (fallback takes SOL without minting) in the same file:

Edit useMintStreamer.ts

Removed 26 lines

Update Todos

Fix BUG 1: require('bs58') crash in useMintStreamer.ts

Fix BUG 10: Fallback takes SOL without minting NFT

Fix BUG 2: Infinite re-subscribe loop in usePvPMatchmaking.ts

Fix BUG 3: Channel re-creation loop in usePvPBattle.ts

Fix BUG 4: Stale isComplete in bot retaliation

Fix BUG 5: syncFromCloud ignores falsy values

Fix BUG 6: Stale player.hp written to DB on reconnect

Fix BUGs 7-8: Client can inject wins/losses/securedIds

Fix BUG 9: Mission state doesn't reset on streamer change

Fix BUG 11: Hardcoded JWT secret fallback

Fix BUGs 12-18: Medium severity bugs

Fix BUGs 19-25: Low severity bugs

Now BUG 2 - fix the infinite re-subscribe loop in matchmaking by memoizing myStats:

Read usePvPMatchmaking.ts

Write usePvPMatchmaking.ts

168 lines

Now fix BUGs 3, 4, 6, 19 in usePvPBattle.ts:

Read usePvPBattle.ts

This is the biggest fix. Let me rewrite usePvPBattle.ts fixing BUGs 3, 4, 6, and 19:

Write usePvPBattle.ts

641 lines

turnLocked is only a hint (unused variable from the original code) — not a bug I introduced. Moving on to BUG 5 now.

Update Todos

Fix BUG 1: require('bs58') crash in useMintStreamer.ts

Fix BUG 10: Fallback takes SOL without minting NFT

Fix BUG 2+12: Infinite re-subscribe loop + retry in usePvPMatchmaking.ts

Fix BUGs 3,4,6,19: PvP battle channel loop, stale isComplete, stale HP, opponent maxHp

Fix BUG 5: syncFromCloud ignores falsy values

Fix BUGs 7-8: Client can inject wins/losses/securedIds

Fix BUG 9: Mission state doesn't reset on streamer change

Fix BUG 11: Hardcoded JWT secret fallback

Fix BUGs 13-14,18: Mission hook medium bugs

Fix BUGs 15-17: Page/sync medium bugs

Fix BUGs 20-25: Low severity bugs