ANDROID · GROWTH · ANTI-ABUSE

A More Secure Referral System

The short version of the ReelBlink referral guide. Why simple referral-code systems are easy to abuse, which Google Play tools make them stronger, and what to check before you ship.

The mistake most referral systems make

Generate a code, friend enters it, code is valid, give reward. Easy to build — but **a valid referral code does not prove a genuine new user was acquired.** People can refer themselves, create multiple accounts, uninstall and reinstall, switch Google accounts, replay the same request, or modify the app.

A referral code should answer only one question: who gets credit? It should not be the proof that the referral is genuine.

What each tool actually answers

TOOL	THE QUESTION IT ANSWERS
Play Install Referrer	Where did this installation come from?
Authentication	Which user or account is this?
Play Integrity API	Does this request come from an expected Play app and device environment?
Server validation	Does this referral satisfy our rules?
Device Recall	Has this device already participated in this one-time promotion?

Google Play has no ready-made referral program. You still build the reward logic, referral UI, rules, backend validation and reward storage yourself — these are just the building blocks.

Define “successful referral” properly

Not a link click. Not an install on its own.

> **Referral attribution** + **Eligible new user** + **Authentication** + **Required onboarding** + **Integrity check** + **Server validation** = **Successful referral**

After detecting attribution, treat it as a **pending** referral. The app may request a reward; the server decides whether it is valid.

Abuse cases and their protection

ABUSE CASE	PROTECTION
Self-referral	Referrer must differ from the referred user
Same account after reinstall	Server-side account history
Same referred user reused	One-time qualification rule
Existing user posing as new	Account creation time, install info, eligible app version
Fake or modified app	Play Integrity
Replay of the same request	Idempotency, unique request IDs, DB constraints
Claiming one reward twice	Atomic server-side claim
Same phone, new account	Device Recall, where available
Link clicked, no conversion	No reward
Install but never qualifies	No reward

Rules that keep it honest

- **Rewards belong to the server.** Never let the app set something like `premiumPackUnlocked = true`. The server checks eligibility and grants the entitlement.
- **Never trust a client claim.** Sending `integrityPassed = true` from the app proves nothing — verify the integrity token server-side.
- **Keep a reward history**, not just a counter. It gives you auditing, duplicate protection and support answers.
- **A stable internal user ID** beats an email address as the main identifier.
- **Temporary failure is not fraud.** On an outage or rate limit, keep the referral pending and retry — do not burn a genuine user's reward.

Device Recall: useful, not required for v1

Play Integrity does not solve the same-phone / new-account case on its own. Device Recall can, by remembering limited abuse-related state for a device. At the time the source guide was written it was available through a beta / controlled-access process with an interest form asking for the developer account, app package, linked Cloud project and the abuse problem being solved.

A strong v1 works without it: install referrer + authentication + Play Integrity + server validation. If Device Recall is not enabled, do not claim same-device abuse is solved.

Linking a Google Cloud project for Play Integrity does not mean your backend has to run on Google Cloud Run. The project is for API configuration and authorization only.

Common mistakes

- Rewarding on link click, or on install with no further verification
- Treating a manual referral code as complete proof
- Trusting user identity or `referralSuccessful = true` sent by the client
- Storing permanent rewards only on the phone
- Letting the app grant itself paid access
- Skipping idempotency or server-side duplicate checks
- Treating infrastructure failure as user fraud
- Claiming Device Recall protection before it is enabled
- Tying referral rewards to Play Store ratings or reviews

Test before you ship

- Genuine referral succeeds
- Self-referral fails
- Same account cannot qualify twice
- Existing user cannot become a fake “new user”
- Invalid attribution is rejected
- A replayed request creates no second reward
- One reward cannot unlock two items
- An invalid reward choice does not consume the reward
- A modified or unrecognised app cannot claim a reward
- A temporary server failure does not permanently burn a referral
- Reward survives reinstall and account restoration
- Account switching does not leak rewards
- Same-device / new-account abuse, if Device Recall is enabled

The main lesson

The hard part is not generating a referral link. It is proving that a genuinely new user was brought by the correct referrer, that the app and the request are legitimate, and that the reward has not already been earned or claimed.

> **Attribution** + **Identity** + **Integrity** + **Server validation** + **Anti-abuse rules**

Condensed from “How I Built a More Secure Referral System for ReelBlink”. ReelBlink: play.google.com/store/apps/details?id=com.classhai.make_blink