ApeShield

Trade Solana. Ape Shielded.

A safety-first, custodial trading bot for the Solana blockchain, delivered entirely through Telegram — with honeypot protection before every trade, provably on-chain fees, and the SHIELD utility token.

Whitepaper v1.1 | Solana · SPL | July 2026

t.me/ApeShieldBot_bot · apeshield.io

Contents

1. Abstract

ApeShield is a **custodial** Telegram trading bot for Solana that puts user safety ahead of every other priority. Users trade SPL tokens, snipe new listings, set stop-loss / take-profit orders, and copy-trade wallets, entirely from a Telegram chat.

Where most trading bots optimise purely for speed and let users buy anything — including tokens engineered to be unsellable — ApeShield runs an automated honeypot and rug-risk assessment **before** a buy is ever offered, and refuses tokens that cannot be sold. Its fees are taken inside the same on-chain transaction as the swap, making them fully auditable. Private keys are protected with envelope encryption and a cloud hardware-security-module master key. The project is accompanied by a fixed-supply utility token, **SHIELD**, which grants holders a reduced trading fee.

This document describes the system's design, its security model, its economics, and — in keeping with the project's core value of honesty — its limitations and risks.

2. The Problem

On-chain trading on Solana is fast and permissionless, but that openness is also its danger. The most common ways retail traders lose money are not market moves — they are structural traps:

- **Honeypots.** Tokens you can buy but never sell, because the contract or liquidity is rigged.
- **Unrenounced authorities.** A deployer who can mint unlimited new supply, or freeze your tokens in your own wallet.
- **Rug pulls.** Liquidity withdrawn by the team, collapsing the price to zero.
- **Opaque bot fees.** Hidden slippage skimming or vague fee accounting that users cannot verify.
- **Custody anxiety.** Handing keys to a bot with no clear way to export or withdraw.
- **Silent failures.** Trades that hang or double-execute during network congestion, leaving funds in an unknown state.

Existing Telegram bots compete almost entirely on execution speed. ApeShield takes a different position: **protect the user first, then be fast.**

Design priorities, in strict order:

1. Security of user funds and private keys — above all else.
2. Correctness of trade execution and fee accounting.
3. Reliability — no lost or stuck transactions; graceful failure.
4. Speed.
5. Feature completeness.

3. The ApeShield Solution

ApeShield is operated through a Telegram bot. Each user is issued a dedicated Solana wallet on first contact. They deposit SOL, and from then on they trade by pasting a token's mint address — which triggers a safety report and a one-tap buy menu — or by using commands for automation, positions, alerts and withdrawals.

Three principles differentiate it:

Protection before purchase

Before a buy button is shown, ApeShield verifies the token is sellable right now, that the mint and freeze authorities are renounced, checks the token program and age, measures the real price impact of an executable route, and reads holder-concentration and liquidity signals. Tokens that fail hard checks are **blocked**, not merely flagged.

Provable honesty

The trading fee is a distinct instruction inside the same transaction as the swap. Anyone can open the transaction on a block explorer and see the exact fee and its destination. The bot never claims to verify something it cannot; where a check relies on a third party or cannot be determined, it says so plainly.

User sovereignty

Users can export their private key or withdraw their funds at any time, through explicit multi-step confirmations. The bot holds keys for convenience, not to trap funds.

4. Security Architecture

Custody of private keys is the highest-value target in the system and is engineered accordingly.

Encryption at rest	Each wallet's Ed25519 secret key is sealed with **AES-256-GCM** under a per-secret data-encryption key (DEK). Every ciphertext is bound to its owner via additional authenticated data (AAD), so a database row copied to another account fails to decrypt.
Master key custody	The DEK is wrapped by a master key held in **AWS KMS** (a FIPS-validated hardware security module). The master key never leaves KMS and never touches the server. Every decryption is logged in AWS CloudTrail and can be revoked instantly.
Decrypt just-in-time	A plaintext key exists in memory only for the moment of signing a transaction, then is zeroized. Keys are never held in long-lived variables, caches, or logs.

Logging discipline	Keys, seeds, and master keys are never logged. A sanitizer middleware redacts key-shaped material as a backstop.
Infrastructure	Dedicated hardened server: SSH key-only access, dual firewalls (network + host) permitting only management and web ports, database and cache bound to localhost and unreachable from the internet. The bot process itself opens no inbound ports.

Breach containment. A stolen database yields only encrypted blobs; a stolen disk yields no master key. To move funds, an attacker would need to simultaneously compromise both the server and the separately-controlled AWS account.

5. Trading Engine & Correctness

Every trade is driven through a persisted state machine that guarantees no double-execution and no fee without a trade:

```
PENDING → SIMULATED → SUBMITTED → CONFIRMED | FAILED
```

Guarantee	Mechanism
Fee atomicity	The fee transfer is composed into the *same* transaction as the swap. There is no trade without the fee, and no fee without the trade.
Pre-flight safety	Every transaction is simulated before signing. A failed simulation is never sent and never charged.
Idempotency	Each order carries a unique key; retries and concurrent workers can never re-execute or double-charge.
Recovery	Confirmation polling with timeout. On blockhash expiry, chain state is re-checked before any retry — eliminating duplicate sends. Timed-out orders are reconciled, never blindly resent.
Concurrency safety	A per-user lock prevents a single user from double-spending by firing two commands at once.
Readable failures	Slippage, insufficient funds, blockhash expiry and RPC outages map to distinct, plain-language messages that always state whether funds moved.

Fee mathematics use integer arithmetic exclusively (no floating point near money), round in the user's favour, and satisfy the invariant that fee plus net input reconstructs the gross input exactly.

6. Features

Wallet & core

- Instant per-user Solana wallet; deposit detection with live notifications.
- Balances and positions with USD value, entry price, and live profit/loss.
- Withdrawals with double destination confirmation and plain-language summary.
- Private-key export behind explicit multi-step confirmation.

Manual trading

- Buy by pasting a mint address; automated honeypot / rug risk report before quoting.
- Preset buy amounts and sell percentages; configurable slippage with sane bounds.
- Transaction simulation before signing; realized fee and net result shown after every trade.

Automation

- **Limit orders** — stop-loss and take-profit price triggers, polled continuously.
- **Sniper** — buys the instant liquidity appears, submitted as an atomic Jito bundle, with a full 5-state lifecycle and an auto-cancel that never spends into a token the shield blocks.
- **Copy trading** — mirror a target wallet within a per-trade cap and a total budget.
- **Price alerts** — notification when a token crosses a chosen on-chain price.

Social & growth

- **PnL & flex cards** — share-ready profit cards carrying the user's referral link.
- **Referrals** — automatic, on-chain fee sharing (see §9).
- Administrator dashboard: volume, fee revenue, active users, activation funnel, error rate.

7. Fee Model

ApeShield charges a transparent fee of **0.75%** (75 basis points) on every executed swap, always taken *in SOL* — carved from the SOL you spend on a buy, or from the SOL you receive on a sell (never in the token itself, so a token that later collapses can never distort or trap the fee). Holders of the SHIELD token qualify for a reduced **0.50%** rate. The fee is:

- **Displayed** on every trade — gross committed, fee, net swapped, amount received.
- **On-chain** — a distinct transfer in the same transaction, auditable by anyone.
- **Recorded** — every fee is written to the database with its trade id and transaction signature.

There are no hidden charges, no slippage skimming, and no misrepresentation of trade outcomes. This is a firm design constraint, not a marketing claim. The team earns from this fee revenue — paid in SOL from real product usage — and holds **no** token allocation (see §8).

8. The SHIELD Token & Distribution

SHIELD is a Solana SPL token whose sole promised function is a trading-fee discount on ApeShield. It is deliberately simple and verifiable — **and every claim below carries its proof link.**

Name / Symbol	ApeShield / **SHIELD**
Contract (mint)	`ECwiqhkYyoksjSDJzF3uZgyRntRk6HjGyMkNh7zfp3tu`
Blockchain	Solana (SPL)
Total supply	**44,550,000,000,000** — 49.5T minted, the 4.95T (10%) team allocation **burned** in full, fixed forever
Decimals	5
Mint authority	**Renounced — no new supply can ever be created**
Freeze authority	**Renounced — no wallet can ever be frozen**
Team allocation	**Zero — the entire 10% deployer holding was burned**
Utility	Hold ≥ 100,000,000 SHIELD in the bot wallet → 0.50% fee instead of 0.75%
Metadata	Name, symbol and logo stored permanently on Arweave

Verify everything: Solscan (token) · RugCheck · Team-allocation burn transaction

Distribution

Allocation	Amount	Status
Liquidity pool (Raydium)	90% of mint (44.55T)	**Seeded at launch; LP tokens burned — liquidity is permanent**
Team / deployer	10% of mint (4.95T)	**Burned in full, 25 Jul 2026 — zero team allocation**
Current total supply	**44.55T**	100% is in the burned liquidity pool or in community hands

Launch disclosure. An unaffiliated wallet acquired roughly 24% of the initial pool in the first minutes of launch and has since partially distributed. This cluster is **not** team-controlled and is publicly traceable on-chain; we track it and disclose it rather than hide it — the same transparency this document applies everywhere.

SHIELD's authorities were renounced at creation, so the token passes ApeShield's own safety checks. The project applies to its own token the same honesty it applies to every other: SHIELD is a utility and community token, **not an investment**, and carries no promise of price or return. The team's alignment is fee revenue from real usage, not a token bag.

9. Referral System

ApeShield's referral payouts are trustless. When a referred user trades, the referrer's share of the fee (25% by default) is transferred to the referrer's wallet **inside the same transaction as the trade itself**. There is no claim process, no payout queue, and no need to trust the operator: the payment is atomic, immediate, and verifiable on-chain. Every user has a personal referral link, embedded automatically in the shareable flex cards.

10. Technology Stack & Infrastructure

Language	TypeScript (Node.js 20, strict mode)
Telegram	grammY framework
Blockchain	@solana/web3.js, Jupiter Aggregator for routing, Jito bundles for snipes
Storage	PostgreSQL (Prisma ORM), Redis for queues, rate-limits and session state
Jobs	BullMQ workers for order execution, limit polling, copy watchers, alerts
Key custody	AWS KMS envelope encryption
Deployment	Dockerised (bot + database + cache) on a hardened dedicated server with dual firewalls, key-only SSH, encrypted daily off-site backups; public site at apeshield.io over HTTPS
Logging	Structured logging with mandatory secret redaction

11. Roadmap

Phase	Focus
Delivered	Wallet & KMS custody; manual trading with pre-trade honeypot/rug checks (sellability, authorities, token age, executable price-impact block gate, holder-concentration, LP-lock status); limit orders; sniper (Jito bundle) with full lifecycle & shield auto-cancel; copy trading; provable SOL-denominated fees on both legs with automatic on-chain referrals; positions & alerts; production deployment; custom domain & HTTPS site (apeshield.io); SHIELD token with burned liquidity and burned team allocation; dedicated fee treasury; activation analytics.
In progress	Post-buy position monitoring (continuous protection after purchase); push-based deposit tracking; expanded admin analytics.

Before public scale	Independent third-party security audit; legal / compliance review; deeper on-chain LP-lock verification recomputed across DEXs and locker programs; lower-latency geyser feeds for sniping.

12. Risk Disclosures

In keeping with the project's core value, these risks are stated plainly.

This is not financial advice

Nothing in this document is an offer, solicitation, or recommendation to buy any token. Cryptocurrency trading involves substantial risk of total loss. Micro-cap tokens can lose their entire value in seconds. Never trade funds you cannot afford to lose.

SHIELD is a utility token, not an investment

SHIELD exists to provide a fee discount. It carries no promise of profit, value, or future price. It should not be purchased as an investment.

Custodial model

ApeShield holds the private key to each user's trading wallet. While keys are protected with envelope encryption and an HSM master key, users should deposit only what they actively trade, and can export their key or withdraw at any time.

Security is engineered, not guaranteed

The system is designed and tested to a high standard, but has **not yet undergone an independent third-party security audit**. No software can be described as perfectly secure. An audit is a prerequisite on the roadmap before the platform is opened at scale to third-party funds.

Regulatory uncertainty

Custodial trading services may constitute regulated money-transmission or virtual-asset-service-provider activity in some jurisdictions, with associated KYC/AML obligations. Users are responsible for compliance with their local laws.

Honest limitations

Liquidity-lock/burn status is read from a third-party provider (RugCheck) and reported honestly — including as "unverified" when it cannot be determined — rather than independently recomputed across every DEX and locker program. Holder-concentration is measured from top-holder data; coordinated-wallet clustering is not claimed, as no reliable low-false-positive signal exists. The sniper detects tradability via routing and is not a block-zero solution. These constraints are documented rather than hidden.

Changelog v1.1: added contract address & verification links; recorded the team-allocation burn and zero-allocation distribution; corrected LP-lock capability; updated roadmap to delivered/in-progress; SOL-both-legs fee model.