neuTracker

Bitcoin Blockchain Forensics & Transaction Tracking

What is neuTracker?

neuTracker is a high-performance Bitcoin blockchain forensics and transaction tracking tool. It indexes the entire Bitcoin blockchain directly from your local block files, builds a fully searchable database of every transaction, address, and public key, and gives you the tools to explore, track, and receive real-time alerts on blockchain activity — all from your own hardware, with no third-party dependencies.

Why is it needed?

The blockchain itself doesn't make this easy.
Bitcoin's raw block files are a flat, append-only binary format. There's no built-in way to look up an address, search by date, or follow the flow of funds between wallets. To do any of that, you need an index — and building one is a massive undertaking.

Third-party blockchain APIs have real limitations:
They're expensive at scale, impose rate limits and throttling, introduce latency on every lookup, and — most critically for forensic work — you're trusting someone else's infrastructure with your queries. Every address you search, every transaction you trace, is visible to the API provider. For law enforcement, compliance, and private investigations, that's a non-starter.

Traditional self-hosted indexers
(Electrum Server, Blockbook, etc.) duplicate enormous amounts of data, require hundreds of gigabytes of additional storage, and can take days or even weeks to complete an initial sync. Many rely on Bitcoin Core's JSON-RPC interface, which was designed for wallet operations — not bulk data extraction.

neuTracker takes a fundamentally different approach.

How it works

Direct block file parsing — no RPC, no middleman.
neuTracker reads Bitcoin's .blk files directly from disk using a custom binary parser. Instead of asking Bitcoin Core to serialize and deserialize data through its JSON-RPC interface (a well-known bottleneck), neuTracker opens the raw files, seeks to exact byte positions, and parses block headers, transactions, inputs, outputs, scripts, and witness data at wire speed.

No data duplication.
The database stores only lightweight references — block heights, transaction hashes, file numbers, and byte offsets. When you look up a transaction in the Explorer, neuTracker seeks directly to that position in the original block file and parses it on the fly. Your block files are the data store. The database is just the index.

Three-phase indexing:

- **Block File Scan** — Every .blk file is opened and each block header is parsed to catalogue every block's location (file number + byte offset).
- **Transaction Scan** — Each block is re-read, this time parsing every transaction to record its hash and file position. This is the most I/O-intensive step.
- **Address / PubKey Scan** — Transaction outputs are decoded to extract destination addresses and public keys. Input scripts and witness data are analyzed to link spending addresses and recover public keys for signature verification.

Each phase creates database indexes in parallel after bulk loading completes, so queries are fast from the moment indexing finishes.

Design choices

.NET 10 — This ain't your grandfather's .NET

Modern .NET compiles to native code with aggressive optimizations — SIMD vectorization, tiered JIT compilation, and hardware intrinsics. In raw throughput benchmarks it regularly matches or beats C++ and Rust for I/O-bound workloads, and leaves Java and Python far behind. neuTracker leverages the Task Parallel Library (TPL) to saturate every available CPU core during indexing. If you've got the cores, neuTracker has the speed.

Avalonia UI — True cross-platform desktop

Avalonia delivers a native-feeling desktop experience on Windows, macOS, and Linux from a single codebase. Unlike Electron-based tools that ship an entire browser, Avalonia renders directly via GPU-accelerated backends (Skia/Direct2D) with minimal memory overhead. It provides the rich control library and data binding of WPF without being locked to Windows, and avoids the inconsistencies and limitations of MAUI, UWP, and WinForms.

PostgreSQL — The right database for the job

Bitcoin's full index involves billions of rows across transactions, addresses, and public keys. PostgreSQL handles this scale with ease: parallel query execution, advanced indexing (B-tree, hash, GIN), efficient bulk loading via its binary COPY protocol, and rock-solid ACID compliance.

SQL Server's licensing costs and resource overhead make it impractical for a dataset this large. SQLite lacks concurrent write support and chokes on multi-gigabyte databases with complex joins. NoSQL databases (MongoDB, etc.) lose the relational query power that makes forensic link analysis possible — following the money across transactions, addresses, and public keys demands joins, aggregations, and range queries that relational SQL was built for.

No RPC

Bitcoin Core's JSON-RPC interface was designed for wallet operations, not bulk data extraction. Every RPC call involves JSON serialization/deserialization, HTTP overhead, and — most critically — Bitcoin Core's own internal locking, which serializes requests. Even on fast hardware, RPC-based indexers are throttled by this bottleneck. neuTracker bypasses it entirely by reading the same binary files that Bitcoin Core writes, using direct file I/O with precise byte-offset seeking.

No ORM — Dapper + Binary COPY

Object-Relational Mappers (Entity Framework, NHibernate) add overhead that matters at billion-row scale: reflection, change tracking, and query generation. neuTracker uses Dapper for lightweight query mapping and PostgreSQL's binary COPY protocol (via NpgsqlBinaryImporter) for bulk inserts — the fastest possible path from parsed data to indexed rows. Indexes are created after bulk loading completes, not during, so inserts aren't slowed by index maintenance.

NBitcoin — Validation only

NBitcoin is an excellent and comprehensive Bitcoin library, but its generality comes with overhead that matters when you're parsing billions of transactions. neuTracker includes a custom BitcoinBlockchainParser tuned for raw speed — minimal allocations, lazy transaction parsing, bounded stream reads, and zero unnecessary copies. NBitcoin is included in the project for address derivation, script analysis, and correctness validation where its rich API is valuable, but the hot path of block file scanning uses the custom parser exclusively.

Performance

neuTracker is engineered for speed at every layer:

- **Parallel file processing** — Each .blk file (or block, depending on the scan phase) is processed on its own thread. All CPU cores are utilized during indexing.
- **Binary COPY bulk loading** — Data flows from parser to database via PostgreSQL's fastest ingestion path. No SQL parsing, no parameter binding, no round-trips per row.
- **Deferred index creation** — Indexes are built in parallel after each scan phase completes, so bulk inserts run at maximum throughput without index maintenance overhead.
- **Lazy transaction parsing** — Block headers are parsed immediately, but transaction data is only deserialized when accessed. This makes the block file scan phase extremely fast.
- **Direct file seeking** — Explorer lookups don't query a copy of the data. They seek to the exact byte offset in the original block file and parse on demand — keeping the database small and lookups fast.
- **XOR-encrypted block file support** — neuTracker transparently handles Bitcoin Core's optional block file obfuscation (XOR cipher) with no measurable performance penalty.

Use the built-in Benchmark Demo to measure your hardware's raw parsing throughput before committing to a full index. It runs the same scan phases with real disk I/O but skips all database writes.

The Explorer

The Explorer is a suite of four interconnected views for investigating blockchain data:

- **Address Explorer** — Search by address, name, or tag. View complete transaction history with inbound/outbound amounts, pagination, and sorting by date or amount. Toggle to the Map view for an interactive fan-out visualization of where funds came from and where they went — expand levels to trace the flow of money across multiple hops. Generate QR codes for any address.
- **Transaction Explorer** — Look up any transaction by its hash, name, date range, or tag. See the full decoded structure: every input (with source address and amount) and every output (with destination address and amount). Navigate directly to related blocks or addresses.
- **Block Explorer** — Browse blocks by height, date range, name, or tag. View all transactions within a block, sorted by position, input/output count, or value.
- **PubKey Explorer** — Search public keys by hex value or name. See every address derived from a given public key — a powerful tool for linking wallets that share a key.

All four explorers are cross-linked: click an address in the Transaction Explorer and jump straight to the Address Explorer. Click a block height and land in the Block Explorer. This lets you follow the money naturally, moving between views as the investigation demands.

Naming and tagging: Any blockchain object — address, transaction, block, or public key — can be given a human-readable name, notes, and tags. Names appear throughout the Explorer, making it easy to recognize known entities (exchanges, mixers, wallets of interest) at a glance. Tags enable filtered searches across your entire dataset.

The Tracker

The Tracker turns neuTracker from a forensic research tool into an active monitoring system.

- **Tracked Objects** — Mark any address, transaction, or public key as tracked. When new blockchain activity involves a tracked object, neuTracker generates an event and classifies it automatically: sent, received, sent to a known address, received from a named entity, public key reuse detected, and more.

- **Known Addresses** — Flag addresses as "known" (exchanges, payment processors, identified wallets). When a tracked address transacts with a known address, the event is classified differently (e.g., "Sent to Known" vs. generic "Sent"), giving you immediate context without manual lookup.
- **Tracking Groups** — Organize tracked objects into groups, each with its own devices (email and MMS endpoints) and alert level thresholds. A device configured for "High" alerts won't be bothered by routine low-severity events, but will be notified immediately when something critical happens.
- **Real-time alerts** — The server pushes notifications over WebSocket the moment a tracked event is detected. Connected clients receive instant visual and audio alerts. Group devices receive email or MMS notifications based on their configured alert level.
- **BackDate** — Retroactively generate tracking events for an address over a historical date range. This processes past transactions as if they were happening live, classifying each event and optionally sending notifications. Ideal for onboarding a new address into your tracking workflow without missing its history.

The API

neuTracker exposes a comprehensive REST API with over 50 endpoints, giving you programmatic access to every feature available in the desktop UI. Query blocks, transactions, addresses, and public keys. Manage tracked objects, tracking groups, and subscriptions. Search by name, tag, date range, or raw identifier.

The Client application is built entirely on this API — it has no direct database access or blockchain file access of its own. Everything you see in the Client UI is fetched over REST, which means any external tool, script, or integration can do the same.

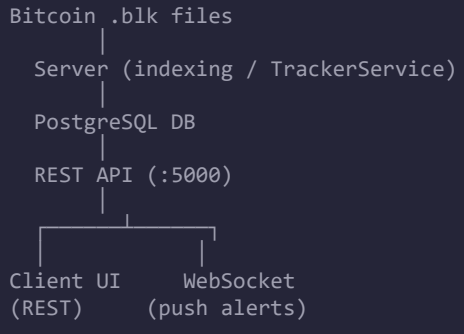
Client / Server architecture

The Server runs on the machine with Bitcoin Core's block files and the PostgreSQL database. It handles all indexing, tracking, and alert dispatch. It also hosts the REST API and WebSocket notification server.

The Client is a lightweight desktop application that connects to the Server remotely. It has no local database, no blockchain files, and no indexing overhead — just a fast, responsive UI backed by the Server's API. Multiple clients can connect to the same server, each authenticated through their Tracking Group credentials.

This separation means investigators can run the Client on any machine — a laptop in the field, a workstation at the office — while the heavy lifting happens on dedicated server hardware.

Data Flow



Security

All queries and data stay on your own infrastructure. No blockchain lookups are sent to external services. Configuration files containing database credentials and SMTP passwords are encrypted at rest using AES-256-GCM authenticated encryption, with the key stored separately in the user's application data directory.

The future

neuTracker's architecture — direct binary file parsing, lightweight database indexing, and a modular scan pipeline — isn't specific to Bitcoin. The same approach can be applied to any blockchain that stores data in parseable file formats. Ethereum and other major chains are on the roadmap. The tooling is built; the next step is extending the parsers.