FAMBF · REFERENCE DOCUMENT

The Bot Safety Checklist.

14 questions that decide whether your trading automation survives a real market.

Pavlo Filianov

Founder, FAMBF — Framework for Automated, Mandate-Backed Finance

~30 MIN READ
14 QUESTIONS · 0–14 SCORE

00 · WHY THIS EXISTS

Introduction.

I started writing this checklist after the third trader I knew personally lost most of an account to a bug in their own bot. None of them lost money to the market. They lost it to something in the automation that should have been caught before the system was ever connected to real capital.

The pattern is always the same. The strategy is fine, the code works in normal conditions, and then something unusual happens. A reconnect at the wrong moment. A duplicate signal. A webhook that arrived twenty minutes late. The bot does something the trader never intended, and by the time anyone looks at the logs, the damage is already done.

This is a list of the things that, in my experience, decide whether automated trading actually survives or whether it slowly destroys an account from the inside.

Some of these are obvious in hindsight. Others only become obvious after you have watched a friend lose six figures. I have tried to keep the whole thing readable in about half an hour, with no special tools needed.

HOW TO USE IT

Score yourself zero or one on each question. Zero means your system doesn't do this, or you're not sure. One means it does, in a way you can describe, point to in the code, and demonstrate if asked.

Total up your score at the end. The scoring guide sits at the back of this document.

Two short notes before you start. Be honest with yourself; this is a tool for you, not a marketing exercise. If you fudge a "yes" on a question you're not certain about, you're optimising for feeling good today and losing money later.

And remember what this checklist actually measures. It looks at the automation, not the strategy itself. A perfect score won't make a bad strategy profitable. What it will do is make sure that when your strategy does work, the automation around it doesn't quietly give the gains back.

QUESTION 01 · DAILY LOSS LIMIT

Does your bot have a hard daily loss limit it cannot override?

WHY IT MATTERS

The most reliable way for automation to wipe out an account is to keep trading on a bad day. Most strategies have a handful of outlier days per year where conditions don't match what the strategy was designed for. Without a hard limit, the bot keeps applying normal logic to abnormal conditions, and losses compound faster than a human would let them.

WHAT GOOD LOOKS LIKE

A configurable threshold (in dollars or percent of account equity) checked before every order submission. When the limit is crossed, the bot halts. From there the behaviour depends on what was pre-defined: open orders may be cancelled, positions may be flattened or held, and an alert is sent to the trader. The halt persists until manually reset or until the next trading session.

COMMON FAILURE

> *The limit exists as a variable in the strategy file but is never actually enforced at the order submission layer. The bot blows past it during a fast market, exactly when it matters most.*

QUESTION 02 · POSITION SIZE CAP

Is there a hard cap on position size, per trade and per symbol?

WHY IT MATTERS

Strategies often size positions dynamically based on volatility or account equity. That works fine until an input misbehaves. A volatility reading that returns zero (because of a data feed gap) can produce an enormous lot size from a dynamic formula, and the exchange will happily fill it.

A simple absolute cap, applied after the strategy's own sizing logic, catches all of these. The cap isn't meant to replace the strategy's sizing. It exists so that when that logic fails for whatever reason, the damage is bounded.

WHAT GOOD LOOKS LIKE

Two ceilings. Per trade: never more than X units in a single order. Per symbol: never more than Y units of total exposure on a single instrument, regardless of how many orders it took to build the position. Both checked at submission, not just in strategy logic. Configurable per instrument, because what's reasonable for a major pair is reckless for a small-cap altcoin.

COMMON FAILURE

> *The cap exists at the strategy level but doesn't account for partial fills, retries, or re-entries after a stop-out. By the third re-entry, actual exposure is several multiples of the stated limit.*

QUESTION 03 · WITHDRAWAL API

Is the API key set up with withdrawal permission explicitly disabled?

WHY IT MATTERS

A trading bot has no legitimate reason to withdraw funds. None. If the system is compromised in any way, whether by a leaked key, an attacker on your machine, or a malicious library update, the only thing standing between the attacker and your money is the permission set on that key.

Disabled withdrawal means the worst case is unwanted positions, which is recoverable. Enabled withdrawal means the funds are gone within minutes.

WHAT GOOD LOOKS LIKE

Every key used by automation is created with withdraw disabled at the exchange. IP-allowlisted where the exchange supports it. Rotated on a known schedule, with 90 days being a reasonable default. Stored in a proper secret manager rather than in a config file or a chat message.

COMMON FAILURE

> *The original key was created with withdrawal enabled "just in case" and now lives in a dotfile on your laptop, the cloud server, and three abandoned project folders.*

QUESTION 04 · DUPLICATE ORDERS

If the bot reconnects after losing connection, will it correctly recognise an order it sent but never got confirmation for?

WHY IT MATTERS

This is the failure mode I see most often in real life. The bot sends an order. The connection drops before the response comes back. The bot reconnects, queries open orders, and the order isn't there, because it already filled. The bot interprets "no open order" as "no order was sent" and submits another. The trader ends up in twice the intended position, with exit logic written for a single unit.

Network drops are not rare. They happen every few hours under normal conditions, more often on a residential connection or a cheap VPS. Any bot running for more than a few weeks will encounter this.

WHAT GOOD LOOKS LIKE

Every order carries a deterministic client_order_id that the exchange can use to refuse duplicates server-side. On reconnect, the bot queries filled orders since the last known timestamp (not just open orders) and reconciles its expected state against the exchange's actual state. Where there's uncertainty, the bot pauses for 30 to 60 seconds before placing anything new.

COMMON FAILURE

> *The bot trusts whatever it sees in "open orders" the moment it reconnects, with no reconciliation against fill history. The first network blip silently doubles the position.*

QUESTION 05 · STALE SIGNALS

Will the bot discard a signal that arrived too late to be useful?

WHY IT MATTERS

Trading signals are valid only within a window. A TradingView alert that says "buy at the open of the 1-hour candle" is meaningful at that open. Three hours later, on a different candle, in a different price regime, the same signal is a randomised trade.

Webhook delays happen all the time. TradingView itself has had multi-minute backlogs during volatile periods. Cloud functions cold-start. Notification services queue. A bot that processes every signal it receives, regardless of age, will eventually take a trade based on instructions from twenty minutes ago.

WHAT GOOD LOOKS LIKE

Every signal carries a timestamp from the source. The bot checks signal age at the moment it's about to act. If the signal is older than a configured threshold (typically seconds for intraday, minutes for swing setups), it's logged, alerted on, and discarded.

COMMON FAILURE

> *The bot processes signals in arrival order with no age check. A 20-minute backlog from a webhook outage produces 20 minutes of stale trades when the queue clears.*

QUESTION 06 · KILL SWITCH

Can you stop the system in under five seconds, from your phone, without logging into the exchange?

WHY IT MATTERS

When something is going wrong with a live bot, every second matters. If the only way to stop it is to SSH into a server and kill a process, or log into the exchange and start cancelling manually, the trader will either freeze or make the wrong move. People don't make good decisions in the first thirty seconds of watching their account behave unexpectedly.

The kill switch needs to be a single deliberate action, accessible from wherever you happen to be.

WHAT GOOD LOOKS LIKE

A simple interface that can be triggered from a phone in under five seconds. A Telegram bot command, a button on a dashboard, an SMS code, anything that doesn't depend on getting into the exchange UI. The kill is logged with a timestamp. After triggering, the bot cannot resume until an explicit manual restart, ideally with confirmation that the trader has reviewed the state.

COMMON FAILURE

> *The "kill switch" requires logging into a cloud console, finding the right service, and stopping a container. The trader fumbles with two-factor authentication while the position moves against them.*

QUESTION 07 · PAPER MODE

When the bot runs in paper mode, is it running the same code path as live?

WHY IT MATTERS

A common pattern is to have paper mode as a separate branch or a different script. Over time, the paper version drifts out of sync with the live code. New features get added to live and forgotten in paper. Bug fixes get applied in one place but not the other.

By the time you want to validate a change in paper, you're testing a system that no longer resembles what's actually trading. A false sense of security is worse than no paper mode at all.

WHAT GOOD LOOKS LIKE

Everything above the execution layer is shared between paper and live: the risk engine, the strategy logic, the order builder, the reconciliation. The only thing that actually differs is the adapter at the very edge. In paper mode it talks to a simulator (testnet, an internal simulator, or replayed live data). In live mode it talks to the real exchange. Switching between the two is a configuration flag, not a code change.

COMMON FAILURE

> *Paper and live are two separate scripts. The last time anyone looked at the paper version was six months ago, and it doesn't even compile against the current data schema.*

QUESTION 08 · DECISION LOGS

Is every decision the bot makes written down with a timestamp?

WHY IT MATTERS

When a bot does something you didn't expect, the first question is always "why". The only way to answer it is from logs, and the logs need to cover more than just orders and fills. Every signal that came in, every filter that ran against it, every risk check it passed or failed, every reason a trade was skipped: all of that needs to be on record. Without it, post-incident analysis is guessing.

Logs also catch slow drift. A filter that has quietly become too tight. Slippage creeping up over weeks. Win rate gradually falling. None of these show on a daily P&L, but all of them matter.

WHAT GOOD LOOKS LIKE

Structured logging (JSON or similar) for every significant event. UTC timestamps with millisecond precision. Clear event types: signal_received, signal_rejected, order_submitted, order_filled, risk_breach, kill_activated. Retained for at least 90 days. Queryable without grepping through gigabytes of text. Even a simple Postgres table with proper indexes works well for this.

COMMON FAILURE

> *The bot writes to a text log file on a server somewhere. Nobody looks until something breaks. By then the file has rotated and the relevant entries are gone.*

QUESTION 09 · API KEY HANDLING

Are API keys encrypted at rest, separated from the codebase, and rotated on a known schedule?

WHY IT MATTERS

API keys are credentials to your money. Most leaks I've heard about weren't sophisticated attacks. They were keys committed to a public Git repository, pasted into a chat to ask for help debugging, left in a screenshot uploaded to support, or stored in a dotfile that got synced to a cloud drive.

Rotation matters because compromise isn't always known. A leaked key sitting unused for months is fine. The same key in the wrong hands during a volatile day costs a lot. Rotation limits the window in which any leak can be exploited.

WHAT GOOD LOOKS LIKE

Keys live in a dedicated secret store: a managed service, a self-hosted vault, or encrypted environment variables provided by the hosting platform. Never in the codebase. Never in plain text on disk. Rotated on a schedule (90 days is sensible), with the rotation process tested ahead of time so it doesn't break the bot when it actually happens.

COMMON FAILURE

> *The trader knows the keys are "somewhere safe" but can't actually point to where, and the last rotation was the day the bot was first deployed.*

QUESTION 10 · INCIDENT PROTOCOL

Do you have a written procedure for when the exchange rejects an order or fails to acknowledge a fill?

WHY IT MATTERS

Exchanges aren't perfectly reliable systems. They reject orders for reasons that don't always make sense. They sometimes accept an order and then report it as rejected. They occasionally lose track of fills and require reconciliation. A bot that assumes the exchange always responds correctly will, sooner or later, end up in a state where its internal view doesn't match the exchange's view.

The incident protocol is the answer to "what does the bot do when reality and its expectations disagree".

WHAT GOOD LOOKS LIKE

A documented list of error scenarios, each with a defined response. For every type of error there should be a clear answer on when it's safe to retry, when the trader needs to be alerted instead of handled silently, and when the whole system should pause until a human confirms what state things are actually in. The protocol is written down, not just implemented in code, so the trader can read it and update it as conditions change.

COMMON FAILURE

> *Error handling is whatever the developer thought to handle in the moment. Unexpected errors crash the process, get auto-restarted, and the bot resumes trading without anyone realising state was reset.*

QUESTION 11 · STAGED LAUNCH

Was this system launched in stages, or was day one already a full-size deployment?

WHY IT MATTERS

Strategies look better in backtests than they perform live. This isn't controversial. Slippage is larger in reality. Fills are slower. Some setups never actually trigger because the data the backtest used was cleaner than the live feed. Every automation has bugs that don't show up until it runs against real exchange behaviour.

A staged launch lets you find these problems with small money instead of full size. The standard sequence is paper first, then shadow mode (live data but no real orders), then micro-live at the smallest size that still tells you something, and finally progressively larger as confidence in the system actually builds.

WHAT GOOD LOOKS LIKE

A pre-agreed launch plan with explicit duration and success criteria for each stage. Position size at micro-live is small enough that a total loss of the test capital wouldn't matter. Promotion from one stage to the next requires a positive review, not just elapsed time. The plan is written down before the bot goes live.

COMMON FAILURE

> *The bot is "tested in paper" for a couple of days, then connected to a live account at normal size because the paper results "looked good". The first real bug shows up on a real position.*

QUESTION 12 · DASHBOARD

Is there a dashboard, separate from the exchange interface, that you actually check on a schedule?

WHY IT MATTERS

Most traders check their bot by logging into the exchange and looking at positions. That tells you what the exchange thinks happened, not what the bot thinks happened. The two views diverging is exactly the kind of thing that should be caught early, and the only way to catch it is to look at the bot's own view of the world independently.

The dashboard is also where slow drift becomes visible. Win rate falling over weeks. Slippage creeping up. Signals being rejected at higher rates because a filter has become too tight. None of these show on a daily P&L, all of them matter.

WHAT GOOD LOOKS LIKE

A simple web interface (doesn't have to be pretty) showing current state: active positions, recent fills, recent signals, recent rejections, current risk utilisation against limits, last kill switch status. Historical view for at least 30 days. Accessible from a phone. Checked on a defined schedule, daily at minimum.

COMMON FAILURE

> *There is no dashboard. The trader checks the exchange app occasionally and trusts the bot is fine because the equity curve is "roughly flat".*

QUESTION 13 · SLIPPAGE

Does the bot track slippage on every fill and pause when slippage exceeds a threshold?

WHY IT MATTERS

Slippage is the difference between the price the bot expected and the price it actually got. In normal conditions, slippage is small and predictable. In abnormal conditions, slippage can become enormous, sometimes orders of magnitude worse than the backtest assumed. The abnormal conditions vary: fast markets, thin liquidity, exchange degradation, regulatory news hitting mid-session.

A bot that ignores slippage will happily keep trading through these conditions, getting filled at progressively worse prices, until the strategy's edge is consumed by execution cost.

WHAT GOOD LOOKS LIKE

Slippage measured for every fill in basis points relative to expected price. Rolling average over a short window. When the average exceeds a threshold (anything above 5x normal is suspicious), the bot pauses trading and alerts. The threshold is configurable per instrument.

COMMON FAILURE

> *Slippage isn't measured at all. The strategy's backtest assumed one basis point of slippage. Live conditions during a stress event produce fifty, and the bot keeps trading.*

QUESTION 14 · AVERAGING CAP

Is the maximum number of averaging or DCA steps enforced in the code, not just in the strategy document?

WHY IT MATTERS

Many strategies use averaging, which means adding to a position as it moves against the original entry. This works fine in a normal market. It also produces some of the worst blow-ups in retail trading, because the strategy's idea of "how many times can I average" is usually larger than what the account can actually support during a serious move.

A strategy document might say "average down up to five times". The code, unless explicitly told otherwise, will average down as many times as the signal fires. During a fast adverse move, that can be ten or twenty times.

WHAT GOOD LOOKS LIKE

A hard counter in the execution layer that tracks averaging steps per position. After the configured limit, no more averaging orders are accepted, regardless of what the signal says. The counter resets only when the position fully closes, not when it partially reduces. The limit is conservative, lower than what the strategy "could" handle in a normal market, because the cap exists for abnormal markets.

COMMON FAILURE

> *The averaging limit lives in the strategy file as a variable the strategy is supposed to respect. During a fast move, the strategy logic doesn't get a chance to check, and the bot keeps averaging into a freefall.*

15 · SCORING

Scoring.

Add up your zeros and ones across the 14 questions. The result tells you where your system stands.

0–5

CRITICAL RISK

The system is one bad day away from a serious loss that has nothing to do with the strategy. Recommend halting live trading until the gaps are closed.

6–9

SIGNIFICANT GAPS

The system will probably work most of the time, but the failure modes that aren't covered are the ones that do the most damage. Worth a focused one- to two-week project to close the unchecked items.

10–12

SOLID

The major risks are covered. Remaining items typically relate to monitoring quality, operational discipline, or edge cases the trader has consciously chosen to accept.

13–14

INSTITUTIONAL-GRADE

The system is built to the standard of what a properly run trading operation should look like. The remaining concern is usually drift over time: operational practices that were good at launch slipping as the system runs for years.

16 · NEXT STEPS

If you want a second opinion.

If you scored below 10 and want a second opinion on which gaps matter most for your specific setup, send me a message. I run paid hardening audits on existing automation. The format is a 30-minute call followed by a written review against this checklist and against your actual code and configuration. The deliverable is a prioritised list: what to fix now, what's fine as-is, what isn't worth touching.

If you scored above 10 and you're thinking about expanding what the automation covers (new markets, additional strategies, heavier risk controls), that's a different conversation, but also one I'm happy to have.

Pavlo Filianov

Founder, FAMBF — Framework for Automated, Mandate-Backed Finance

25 years trading the markets, the first 13 partly through a licensed brokerage I founded in Ukraine, and the last decade mostly in crypto. I build automation for self-directed traders who want their rules to actually run on their own account.

→ fambf.com

→ pavlo@fambf.com

→ Book a 30-minute call

→ LinkedIn