The Crypto Pre-Pump Playbook

A Trader's Guide to Finding Explosive Gains and Navigating Extreme Risk

By MT

Copyright © 2025

All rights reserved. This book or any portion thereof may not be reproduced or used in any manner whatsoever without the express written permission of the publisher except for the use of brief quotations in a book review or scholarly journal.

> **Disclaimer:** This book is intended for informational and educational purposes only and is not a recommendation to buy or sell any security. Trading involves substantial risk of loss and is not suitable for every investor. The author is not a financial advisor. All information presented in this book is based on the author's personal experience and research and should not be considered financial advice.
>
> The strategies discussed herein involve targeting highly volatile and illiquid assets, which are the primary targets for market manipulation, including "pump-and-dump" schemes and "rug pulls." The tools and techniques described can identify signals that may represent either legitimate market momentum or fraudulent activity. Distinguishing between the two requires significant due diligence beyond the scope of any automated tool.
>
> Always consult with a qualified financial professional before making any investment decisions. Past performance is not indicative of future results. All trading strategies should be thoroughly tested in a simulated environment before risking real capital. You are solely responsible for your own trading decisions and any resulting losses.

Introduction: The Unseen Edge in Volatile Markets

Forget the fancy algorithms, the mainstream financial news, and the endless search for "insider tips." The true edge in volatile markets—the kind that generates 800%, 1500%, or even 2000%+ gains in days or hours—isn't about being first to *hear* the news. It's about being first to *see* the unusual activity that precedes the news, the hype, and the parabolic price surge.

This isn't about chasing pumps. This is about spotting the early, subtle whispers of institutional accumulation or coordinated interest before the masses even know a stock exists. It's about identifying the "pre-pump" phase.

You've seen the charts for tokens like CHILLHOUSE-USD. You know these moves exist. They don't just explode out of nowhere; they show their hand through volume and on-chain data. This playbook is designed to deconstruct that phenomenon and provide a rigorous, disciplined framework to capitalize on it.

However, a stark warning is necessary before we proceed. The very signals that flag a potential organic breakout are often functionally identical to the signals of a maliciously orchestrated pump-and-dump scheme. The low-liquidity environments we seek for explosive gains are the preferred hunting grounds for scammers and predatory trading bots. Therefore, this playbook is not a simple "how-to" guide; it is a dual-purpose manual:

1. To provide a high-level framework for identifying potential high-momentum opportunities.
2. To instill an even more critical understanding of the extreme risks involved and the due diligence required to avoid catastrophic loss.

This is not a get-rich-quick scheme; it's a get-smart-quick strategy for exploiting market inefficiencies while navigating a minefield of potential fraud. It demands discipline, learning, and decisive action. For those who apply its principles with the gravity they deserve, the rewards can be exponential.

Are you ready to unlock the secrets of the pre-pump? Let's dive in.

Part 1: The Foundation – Mindset & Market Mechanics

Chapter 1: The Alpha Mindset: Conquering Your Inner Demons

Before any chart is analyzed or any trade is placed, the most critical battle must be won: the one within your own mind. Trading, especially in volatile markets, is as much a psychological game as it is a technical one. The difference between consistently profitable traders and those who perpetually blow up their accounts often comes down to mental fortitude.

The Core Philosophy: Risk-First, Profit-Second The foundation of the Alpha Blueprint is simple but non-negotiable: you are not a profit-seeker; you are a professional risk manager. Your primary job is to protect your capital. Profits are simply the byproduct of well-managed risk. This mindset shift is the single most important step toward longevity and success.

Understanding FOMO, Greed, and Fear These are the three horsemen of the trading apocalypse. They are diametrically opposed to rational decision-making.

- **Fear:** Causes you to hesitate on valid entries, cut winning trades too early, or hold losing trades far too long, hoping they "come back." The antidote to fear is a plan. Your stop-loss is your ultimate fear antidote because it defines your maximum risk

  before you even enter the trade.
- **Greed:** Causes you to chase parabolic moves, over-leverage your positions, or refuse to take profits, hoping for "just a little more." This is how winners become losers. The antidote to greed is a pre-defined exit strategy. Taking profits systematically is how you get paid.
- **FOMO (Fear Of Missing Out):** This is a toxic combination of fear and greed. It's the impulse to jump into a trade simply because it's moving, without a proper setup or plan. FOMO is the reason most traders buy at the top, providing the exit liquidity for smarter money. The antidote is patience and discipline, waiting for *your* setup, not just *any* setup.

Chapter 2: Anatomy of a Pump: Signal vs. Deception

To hunt for pre-pump signals, you must first understand the anatomy of a pump itself, in both its legitimate and illegitimate forms. The "Pump Radar" tool's most critical blind spot is its functional inability to discern the intent behind the on-chain data it presents.

The Anatomy of a Pump-and-Dump Scheme A classic pump-and-dump is a form of securities fraud that unfolds in four phases:

1. **Accumulation:** A group of manipulators secretly accumulates a large position in a low-liquidity token at a very low price.
2. **The Pump:** The manipulators launch a coordinated social media campaign (Telegram, X, Discord) to create artificial hype and FOMO. They make coordinated buys to initiate the price surge, which gets the token onto scanners.

3. **The Dump:** As outside investors, lured by the hype, begin to buy in, the original manipulators "dump" their entire position, selling into the artificially created demand.
4. **The Crash:** The massive, sudden sell-off overwhelms the market, causing the price to plummet, often by over 90%, in minutes.

The Radar's Dilemma: Breakout Detector or Victim Finder? The central, unresolvable dilemma is that the on-chain signals of a pump-and-dump are functionally identical to the signals our tools are programmed to detect: a rapid price increase, a surge in volume, and low liquidity. A tool operating on purely quantitative data cannot perform the qualitative due diligence needed to distinguish a legitimate project from a scam. Therefore, it is probable that such a tool often functions as a "victim-finder," identifying assets being actively manipulated and presenting them as opportunities.

Advanced Threats: Rug Pulls and Honeypots Beyond pump-and-dumps, the DeFi space is rife with even more direct forms of theft:

- **Rug Pull (Liquidity Stealing):** Developers create a token, pair it with a valuable crypto like ETH in a liquidity pool, and once investors have swapped their valuable crypto into the pool, the developers withdraw it all, leaving behind a worthless token.
- **Honeypot (Disabled Selling):** A malicious function is written into the smart contract that prevents investors from selling the token. Capital can flow in, but it cannot flow out (except for the developers). A simple test of buying and immediately selling a small amount can often detect this.

Chapter 3: Decoding the Pre-Pump Signals

Understanding the risks, we can now analyze the core signals that precede explosive moves. Each is a double-edged sword that must be interpreted with caution.

- **Volume: The True Voice of the Market:** The core principle is that **Volume Precedes Price**. A stock sitting dormant that suddenly trades multiples of its average volume is signaling that significant capital is flowing in. We must learn to read the "volume clues":
    - **Absorption Volume:** High volume that occurs *without* a significant price increase, often in a tight range. This is a highly bullish signal that suggests large players are accumulating a position without driving up the price.

    - **Climactic Volume:** A parabolic surge in volume at the *peak* of a move. This is often the point of maximum FOMO for retail and the point where smart money is distributing shares. It's a major exit signal.
- **Liquidity: The Double-Edged Sword:** Low liquidity is the amplifier. It's why small amounts of capital can cause exponential price increases. Our tools are explicitly designed to find these illiquid markets by using a

  `maxLiquidity` filter. However, this creates two massive risks:

    - **Price Impact & Slippage:** In illiquid markets, your own buy order can significantly drive up your average execution price. The price you see is not the price you get.
    - **The Profit-Taking Paradox:** The same illiquidity that makes a token easy to pump makes it nearly impossible to exit a large position without crashing the price yourself. The very act of taking profit can destroy the profit itself.
- **Token Age & Float:**
    - **Float (for stocks):** A low float (under 20-50 million shares) means fewer shares are available, amplifying the effect of buying pressure.
    - **Age (for crypto):** New tokens (hours or days old) are prime targets for manipulation as they lack a trading history and established holder base.

Part 2: The Arsenal - Tools for Detection

Chapter 4: The Basic Scan - Using Free Web Tools

You don't need expensive terminals to begin. Free, readily available tools are powerful enough to identify initial opportunities.

TradingView (for Stocks & Crypto) TradingView is your primary charting and screening platform. The free version is incredibly robust.

- **Crafting Your "Pre-Pump" Scan Filters:**
    - **Market/Exchange:** NASDAQ, NYSE, AMEX for stocks; Binance, Coinbase, KuCoin for crypto.
    - **Price:** $0.0001 - $5.00 for maximum explosiveness.

- **Average Volume:** >500k shares (stocks), >$100k (crypto) to ensure minimal liquidity.
- **Relative Volume (RV):** This is your primary trigger. Filter for RV > 5.0 to find stocks with extreme, concentrated interest today.
- **Change %:** Filter for 5% - 15% gain. This range catches stocks that have started their move but are still early enough for massive upside.
- **Float (stocks):** Filter for Float < 50 million shares.

DexScreener (for Crypto) The "Real-Time Crypto Pump Radar" tool analyzed previously is built on the DexScreener API. You can replicate its core logic directly on their website or by using their API.

- **API Query Logic:** The tool constructs a query using parameters like `p24h` (24h price change), `v24h` (24h volume), and `liq` (liquidity). A typical query looks for:

  `p24h > 5 and v24h > 10000 and liq > 5000 and liq < 50000`.

- **The "Confidence Score":** We can enhance this basic scan by adding a "Confidence Score" to rank the results. This is a weighted average of several factors:
  1. **Price Momentum (30% weight):** Higher 24-hour change gets a higher score.
  2. **Volume-to-Liquidity Ratio (30% weight):** A high ratio indicates strong turnover and interest relative to the pool size.
  3. **Liquidity Score (20% weight):** Scores higher for liquidity in the "sweet spot" (e.g., $50k-$200k), which is low enough to move but high enough to trade.
  4. **Token Age (20% weight):** Newer tokens (under 24 hours old) receive a higher score as they are more prone to explosive, early-stage moves.

Chapter 5: The Advanced Scanner - Your Digital Eye with Python

While web tools are excellent for starting, building your own scanner in Python provides the ultimate power and customization. The `CryptoPumpGuard` is a comprehensive system designed to implement the principles of this playbook. It integrates market data analysis, social media monitoring, risk management, and trade execution.

Core Components of `CryptoPumpGuard`:

- **`AdvancedPumpDetector`:** The main class that fetches market data from Binance, calculates advanced features (VWAP, volatility), and uses machine learning models (`IsolationForest`) to detect price and volume anomalies.
- **`TelegramMonitor`:** Uses the `telethon` library to connect to specified Telegram channels, listen for keywords ("pump," "moon," "100x"), and extract ticker symbols from messages in real-time. This provides the crucial social sentiment layer.
- **`TradingSignalGenerator`:** Takes the alerts from the detector and formulates a complete, risk-managed trade plan. It calculates position size based on the 2% rule, and sets dynamic stop-loss and take-profit levels based on volatility and risk-reward ratios.
- **`PumpBacktester`:** Allows you to test your strategies on real historical data from Binance, providing performance metrics like win rate, profit factor, and max drawdown to validate your approach before risking capital.

The full, finalized code for this entire system is provided in **Appendix B**.

Chapter 6: The Alert Advantage - Never Miss a Signal

Finding a signal is useless if you're not there to act on it. A robust alert system is your automated sentry, monitoring the market 24/7.

- **TradingView Alerts:** Use for chart-based alerts. Set alerts for price crossing a key resistance level or for volume exceeding a specific threshold.
- **Automated Python Alerts to Discord/Telegram:** This is the most powerful method. By integrating a webhook URL into your Python scanner (as shown in the `CryptoPumpGuard` code and the `send_discord_alert` function in the Appendix), your custom script can send richly formatted, real-time alerts directly to your private channels.

Part 3: The Execution - The Risk-First Blueprint

Chapter 7: The Pre-Trade Gauntlet: A Non-Negotiable Checklist

Before committing a single dollar, every potential opportunity must pass through this two-stage checklist. This forces discipline and moves you from a quantitative signal to a qualitative decision.

Stage 1: The Quantitative Check (The Scanner's Job)

- [ ] **Unusual Volume?** Is Relative Volume > 5x? Is volume suggesting accumulation?
- [ ]

  Correct Float/Supply? Is it a low-float stock or a low-supply crypto?

- [ ]

  Strong Price Action? Is the price breaking out or showing strong momentum (e.g., >5% gain)?

Stage 2: The Qualitative Check (Your Job - The MOST Critical Step) This is the due diligence that automated tools cannot perform.

- [ ] **Verify the Project and Team:** Move beyond the ticker. Is the team public and verifiable on platforms like LinkedIn? Anonymous teams are a massive red flag.
- [ ]

  Scrutinize the Whitepaper: Does it present a coherent solution to a real problem, or is it a vague collection of buzzwords?

- [ ] **Analyze Community Sentiment:** Join their Discord/Telegram. Is the discussion healthy and technical, or is it pure "to the moon" hype? Are critical questions being deleted or users banned? The latter is a sign of a scam in progress.
- [ ] **On-Chain Forensics:** Use a block explorer (like Etherscan) to verify key details:
  - **Holder Distribution:** Do the top 10 wallets hold >50% of the supply? This is a major dump risk.
  - **Liquidity Lock/Burn:** Is the liquidity provably locked in a time-lock contract or burned? If not, it could be a rug pull. This is one of the most important checks.
  - **Smart Contract Audit:** Has the contract been audited by a reputable firm (e.g., CertiK)? An unaudited contract is a huge risk.
- [ ] **Execute a Test Transaction:** Before investing significant capital, make a small test purchase and, crucially, an immediate test sale. This verifies the contract is not a honeypot that prevents selling.

Chapter 8: Precision Entries and Ironclad Exits

Entry Strategies

1. **The Initial Breakout:** Entering as the price breaks a significant resistance level on a surge of high relative volume. This is an aggressive entry for high-conviction setups.
2. **The Healthy Pullback:** After an initial surge, wait for the price to pull back to a key support level (like the 9-EMA or VWAP) on *decreasing* volume. This is a safer, more conservative entry.

The Golden Rule: Position Sizing & The 2% Rule This is the most important aspect of the blueprint.

Never risk more than 1-2% of your total trading capital on any single trade.

- **Formula:** `Shares to Buy = (Account Size * Risk %) / (Entry Price - Stop-Loss Price)`
- **Example:**
    - Account Size: $10,000
    - Risk: 2% (Max Loss = $200)
    - Entry Price: $5.00
    - Stop-Loss Price: $4.75 (Risk per share = $0.25)
    - **Position Size = $200 / $0.25 = 800 shares.**

This formula ensures that if your stop-loss is hit, you only lose your predefined, acceptable amount.

Exit Strategies

1. **The Hard Stop-Loss:** Your stop-loss is not a suggestion; it is an order. Always place it below a significant technical level. If it's hit, your trade thesis is invalidated. Exit without emotion.
2. **Scaling Out (Partial Profit Taking):** This is the most effective way to manage explosive moves.
    - **Target 1 (2R):** Sell 25-50% of your position when the price hits twice your initial risk.

    - **Move Stop to Breakeven:** Immediately move your stop-loss for the remaining shares to your entry price. The rest of the trade is now risk-free.
    - **Target 2 (3R+):** Sell another portion at a higher target.
    - **Let the Runner Ride:** Use a trailing stop on the final portion to capture the maximum upside.
3. **The Trailing Stop:** An order that automatically moves your stop-loss up as the price rises, locking in profits. A 10-15% trail is common for volatile assets.
4. **Exhaustion/Climactic Volume Exit:** If you see a massive, parabolic spike in price on the highest volume of the move, it's often an exhaustion signal. This is the time to sell aggressively, as it often marks the top.

Part 4: Mastery & Beyond

Chapter 9: The Trader's Journal: Your Path to Consistency

If there is one habit that separates professional traders from amateurs, it is diligent journaling. Your journal is where you conduct a post-mortem on every trade, identify your unique edge, and unmask your emotional biases.

What to Track:

- **Quantitative Data:** Ticker, Strategy, Entry/Exit Price, Shares, P&L, R-Multiple.
- **Qualitative Data (Most Important):**
    - **Setup/Catalyst:** *Why* did you take the trade? (e.g., "Volume spike + insider buy").
    - **Mistake/Win Lesson:** *What did you learn?* (e.g., "Held through dip because of low volume" or "Chased entry due to FOMO").
    - **Emotional State:** How did you feel before, during, and after? (e.g., "Confident," "Anxious," "Greedy").
    - **Chart Screenshot:** A visual record of your entry and exit points.

A link to a free, comprehensive Google Sheets trade journal template is provided in the Appendices.

Chapter 10: Monetizing Your Edge (Ethically)

As you gain expertise, your knowledge becomes a valuable asset. You can build a brand and additional income streams around your skill, but it must be done ethically.

- **Content Strategy:** Provide value first. Create content on platforms like Twitter/X or TikTok that teaches your process. Use templates to break down your trades, explaining the "why" behind your decisions.
- **The Revenue Funnel:**
    - **Top (Free):** Give away immense value through social media content and a free guide (like this playbook).
    - **Middle (Low-Cost):** Offer a premium newsletter with watchlists or an in-depth webinar.
    - **Bottom (Premium):** Create an exclusive Discord community, mentorship group, or a comprehensive video course.
- **Ethical Foundation:**
    - **Full Disclaimers:** Always state that you are not a financial advisor and that trading is risky.
    - **Transparency:** Be honest about both wins and losses.
    - **Teach, Don't Tell:** Empower your audience to make their own decisions; never give direct financial advice.

Conclusion: The Journey of the Forever Student

The Alpha Blueprint is not a static document; it is a dynamic framework. The market is a relentless and evolving arena. To survive and thrive, you must adopt the mindset of a "Forever Student."

- **Backtest & Forward-Test:** Use historical data and paper trading to validate any new strategies or changes to your rules before risking real capital.
- **Adapt, Don't Abandon:** When a setup stops working, analyze why. Don't throw out the entire blueprint; refine it.
- **Embrace Discipline:** Your plan is your shield against the market's emotional chaos. Stick to it.

The journey to trading Alpha is a marathon of consistent effort, continuous learning, and unwavering discipline. The blueprint is laid out. The tools are explained. The risks are clear. Now, it's up to you to execute.

Go forth, trade wisely, manage your risk, and unlock your true potential.

Appendices

Appendix A: Glossary of Key Terms

- **Alpha:** Excess return of an investment relative to a benchmark.
- **Climactic Volume:** A parabolic volume spike at the peak of a move, often signaling exhaustion.
- **Dark Pools:** Private exchanges for large institutional trades.
- **Float:** The number of a company's shares available for public trading.
- **Honeypot:** A malicious smart contract that prevents investors from selling a token.
- **Liquidity:** The ease with which an asset can be bought or sold without affecting its price.
- **Position Sizing:** Determining the number of shares to trade based on risk tolerance and account size.
- **R-Multiple:** A measure of risk-to-reward, showing how many multiples of your initial risk were gained or lost.
- **Relative Volume (RV):** A ratio comparing the current trading volume to the average volume for the same period.
- **Rug Pull:** An exit scam where developers abandon a project and abscond with investor funds.
- **Slippage:** The difference between the expected price of a trade and the price at which it is actually executed.
- **Stop-Loss:** An order placed to sell a security when it reaches a certain price to limit losses.

- **VWAP (Volume Weighted Average Price):** The average price a security has traded at throughout the day, weighted by volume.

Appendix B: Full Python Code Library for The Advanced Scanner (`CryptoPumpGuard`)

This is the complete, finalized code for the `CryptoPumpGuard` system.

Python

```
# crypto_pump_guard.py
# Final Implementation - V2.0

import os
import re
import time
import json
import asyncio
import nest_asyncio
import numpy as np
import pandas as pd
import networkx as nx
import matplotlib.pyplot as plt
from datetime import datetime, timedelta

# Binance and Machine Learning Imports
from binance.client import Client
from binance.exceptions import BinanceAPIException
from sklearn.ensemble import IsolationForest
from sklearn.preprocessing import StandardScaler

# Telegram Imports
from telethon import TelegramClient, events
from telethon.tl.types import PeerChannel

# Apply nest_asyncio for compatibility with environments like Jupyter
nest_asyncio.apply()

class TelegramMonitor:
    """Monitors Telegram channels for pump signals"""
    def __init__(self, api_id, api_hash, channel_ids, keywords):
        self.api_id = api_id
        self.api_hash = api_hash
        self.channel_ids = channel_ids
        self.keywords = keywords
        self.client = None
        self.monitoring_task = None
        self.detected_signals = []
```

```
        self.session_file = 'telegram_session.session'

    async def start(self):
        """Start the Telegram monitoring client"""
        self.client = TelegramClient(
            self.session_file,
            self.api_id,
            self.api_hash
        )
        try:
            await self.client.start()
            print("✅ Telegram client started")
            self.client.add_event_handler(self.handle_new_message,
events.NewMessage)
            for channel_id in self.channel_ids:
                try:
                    entity = await
self.client.get_entity(PeerChannel(channel_id))
                    await self.client.join_channel(entity)
                    print(f"✅ Joined channel: {entity.title}")
                except Exception as e:
                    print(f"⚠️ Failed to join channel {channel_id}:
{str(e)}")
            self.monitoring_task =
asyncio.create_task(self.run_monitoring())
            return True
        except Exception as e:
            print(f"❌ Telegram client failed to start: {str(e)}")
            return False

    async def run_monitoring(self):
        """Main monitoring loop"""
        print("🚀 Starting Telegram monitoring...")
        await self.client.run_until_disconnected()

    async def handle_new_message(self, event):
        """Process incoming Telegram messages"""
        message = event.message
        if message and message.text:
            content = message.text.lower()
            if any(re.search(rf"\b{kw}\b", content, re.IGNORECASE)
for kw in self.keywords):
                tickers = re.findall(r'\$[A-Z]{3,6}',
message.text.upper())
                try:
                    chat = await event.get_chat()
                    channel_name = chat.title
                except:
                    channel_name = "Unknown"
```

```
                signal = {
                    'timestamp': datetime.now(),
                    'channel': channel_name,
                    'message': message.text,
                    'tickers': tickers,
                    'url':
f"https://t.me/c/{event.chat_id}/{message.id}"
                }
                self.detected_signals.append(signal)
                print(f"🚨 Pump signal detected in {channel_name}:
{message.text[:50]}...")

    async def stop(self):
        """Stop the monitoring client"""
        if self.client:
            await self.client.disconnect()
            print("🛑 Telegram client stopped")

class TradingSignalGenerator:
    """Generates and manages risk-controlled trading signals"""
    def __init__(self, account_size=10000, max_risk_per_trade=0.01,
rrr_threshold=3.0):
        self.account_size = account_size
        self.max_risk_per_trade = max_risk_per_trade
        self.rrr_threshold = rrr_threshold
        self.trade_history = []
        self.performance_metrics = {
            'total_trades': 0, 'winning_trades': 0, 'losing_trades':
0, 'win_rate': 0.0,
            'profit_factor': 0.0, 'max_drawdown': 0.0,
'sharpe_ratio': 0.0,
            'equity_high': account_size, 'equity_low': account_size
        }

    def generate_signal(self, detection):
        """Generate risk-managed trading signal"""
        if detection['risk_score'] < 75:
            return None
        symbol = detection['symbol']
        entry_price = detection['price']
        stop_loss, take_profit = self.calculate_levels(detection)
        position_size = self.calculate_position_size(entry_price,
stop_loss)
        confidence = self.calculate_confidence(detection)
        if position_size <= 0:
            return None
        signal = {
            'symbol': symbol, 'timestamp': datetime.now(),
'signal_type': 'LONG',
```

```
            'entry_price': entry_price, 'stop_loss': stop_loss,
'take_profit': take_profit,
            'position_size': position_size, 'risk_score':
detection['risk_score'],
            'confidence': confidence, 'rrr': (take_profit -
entry_price) / (entry_price - stop_loss),
            'status': 'PENDING', 'profit': 0.0, 'roi': 0.0
        }
        return signal

    def calculate_levels(self, detection):
        """Calculate dynamic stop-loss and take-profit levels"""
        price = detection['price']
        volatility = detection.get('volatility', 0.05)
        base_sl_distance = price * volatility * 2
        risk_factor = max(0.5, 1.0 - (detection['risk_score'] - 75) /
25)
        sl_distance = base_sl_distance * risk_factor
        tp_distance = sl_distance * self.rrr_threshold
        stop_loss = price - sl_distance
        take_profit = price + tp_distance
        if stop_loss >= price:
            stop_loss = price * 0.99
        return stop_loss, take_profit

    def calculate_position_size(self, entry_price, stop_loss):
        """Calculate position size based on risk parameters"""
        risk_per_share = entry_price - stop_loss
        if risk_per_share <= 0:
            return 0
        max_risk_amount = self.account_size * self.max_risk_per_trade
        position_size = max_risk_amount / risk_per_share
        return min(position_size, self.account_size * 0.1)

    def calculate_confidence(self, detection):
        """Calculate trade confidence based on multiple factors"""
        confidence = min(95, detection['risk_score'] * 0.8)
        if detection.get('social_signal', 0):
            confidence = min(99, confidence + 10)
        volatility = detection.get('volatility', 0)
        if volatility > 0.1:
            confidence = max(50, confidence - 10)
        return confidence

    def execute_trade(self, signal, current_price):
        """Execute trade based on signal and current market
conditions"""
        if signal['status'] != 'PENDING' or current_price >
signal['entry_price'] * 1.05:
```

```
            return None
        signal.update({
            'status': 'ACTIVE', 'entry_time': datetime.now(),
'current_price': current_price,
            'peak_price': current_price, 'drawdown': 0.0
        })
        self.trade_history.append(signal)
        self.performance_metrics['total_trades'] += 1
        return signal

    def monitor_trade(self, signal, current_price):
        """Monitor active trade and manage exits"""
        if signal['status'] != 'ACTIVE':
            return None
        signal['current_price'] = current_price
        if current_price > signal['peak_price']:
            signal['peak_price'] = current_price
        current_drawdown = (signal['peak_price'] - current_price) /
signal['peak_price']
        signal['drawdown'] = max(signal['drawdown'],
current_drawdown)

        if current_price <= signal['stop_loss']:
            self.close_trade(signal, current_price, 'STOP_LOSS')
        elif current_price >= signal['take_profit']:
            self.close_trade(signal, current_price, 'TAKE_PROFIT')
        elif current_drawdown > 0.1 and current_price <
signal['peak_price'] * 0.9:
            self.close_trade(signal, current_price, 'TRAILING_STOP')
        return signal

    def close_trade(self, signal, exit_price, exit_reason):
        """Close trade and record results"""
        profit = (exit_price - signal['entry_price']) *
signal['position_size']
        signal.update({
            'status': 'CLOSED', 'exit_time': datetime.now(),
'exit_price': exit_price,
            'exit_reason': exit_reason, 'profit': profit,
            'roi': (exit_price - signal['entry_price']) /
signal['entry_price'] * 100
        })
        self.account_size += profit
        if profit > 0:
            self.performance_metrics['winning_trades'] += 1
        else:
            self.performance_metrics['losing_trades'] += 1
        self.update_performance_metrics()
```

```
    def update_performance_metrics(self):
        """Update performance metrics based on trade history"""
        total_trades = self.performance_metrics['total_trades']
        if total_trades > 0:
            self.performance_metrics['win_rate'] =
self.performance_metrics['winning_trades'] / total_trades * 100
        total_profit = sum(t['profit'] for t in self.trade_history if
t.get('profit', 0) > 0)
        total_loss = abs(sum(t['profit'] for t in self.trade_history
if t.get('profit', 0) < 0))
        if total_loss > 0:
            self.performance_metrics['profit_factor'] =
total_profit / total_loss
        if self.account_size >
self.performance_metrics['equity_high']:
            self.performance_metrics['equity_high'] =
self.account_size
        if self.account_size <
self.performance_metrics['equity_low']:
            self.performance_metrics['equity_low'] =
self.account_size
        self.performance_metrics['max_drawdown'] = (
            (self.performance_metrics['equity_high'] -
self.performance_metrics['equity_low']) /
self.performance_metrics['equity_high'] * 100
        )

    def save_trade_history(self, filename='trade_history.json'):
        """Save trade history to JSON file"""
        try:
            with open(filename, 'w') as f:
                json.dump(self.trade_history, f, default=str,
indent=2)
            print(f"💾 Trade history saved to {filename}")
        except Exception as e:
            print(f"❌ Failed to save trade history: {str(e)}")

class AdvancedPumpDetector(TelegramMonitor):
    """Advanced cryptocurrency pump detection system"""
    def __init__(self, api_key, api_secret, watchlist,
telegram_api_id, telegram_api_hash, telegram_channels,
account_size=10000):
        super().__init__(
            api_id=telegram_api_id, api_hash=telegram_api_hash,
channel_ids=telegram_channels,
            keywords=["pump", "moon", "100x", "rocket", "buy wall",
"dump", "whale alert"]
        )
        self.client = Client(api_key, api_secret)
```

```
        self.watchlist = watchlist
        self.historical_data = {}
        self.model = self.load_detection_model()
        self.trader =
TradingSignalGenerator(account_size=account_size)
        os.makedirs('data', exist_ok=True)

    def load_detection_model(self):
        """Load or create detection models"""
        return {
            'volume_model': IsolationForest(n_estimators=150,
contamination=0.005),
            'price_model': IsolationForest(n_estimators=100,
contamination=0.01),
        }

    def fetch_market_data(self, symbol, interval='1m', lookback=24):
        """Fetch market data with robust error handling"""
        try:
            klines = self.client.get_klines(symbol=symbol,
interval=interval, limit=lookback * 60)
            df = pd.DataFrame(klines, columns=[
                'timestamp', 'open', 'high', 'low', 'close',
'volume', 'close_time',
                'quote_asset_volume', 'trades', 'taker_buy_base',
'taker_buy_quote', 'ignore'
            ])
            numeric_cols = ['open', 'high', 'low', 'close', 'volume',
'quote_asset_volume']
            df[numeric_cols] = df[numeric_cols].apply(pd.to_numeric,
errors='coerce')
            df['timestamp'] = pd.to_datetime(df['timestamp'],
unit='ms')
            df['returns'] = df['close'].pct_change()
            df['volatility'] = df['returns'].rolling(10).std()
            df['vwap'] = (df['quote_asset_volume'].cumsum() /
df['volume'].cumsum())
            self.historical_data[symbol] = df
            return df
        except BinanceAPIException as e:
            print(f"⚠️ Binance API error for {symbol}:
{e.status_code} {e.message}")
        except Exception as e:
            print(f"⚠️ Error fetching data for {symbol}: {str(e)}")
        return pd.DataFrame()

    def detect_pump_signals(self, symbol):
        """Comprehensive pump detection with multiple models"""
        df = self.fetch_market_data(symbol)
```

```
        if df is None or df.empty or len(df) < 10:
            return None

        # Feature Engineering and Anomaly Detection
        features = df[['close', 'volume', 'returns',
'volatility']].copy().dropna()
        if len(features) < 10: return None

        scaler = StandardScaler()
        scaled_features = scaler.fit_transform(features)

        volume_anomalies =
self.model['volume_model'].fit_predict(scaled_features[:,
1].reshape(-1, 1))
        price_anomalies =
self.model['price_model'].fit_predict(scaled_features[:,
0].reshape(-1, 1))

        latest = features.iloc[-1]
        risk_score = self.calculate_risk_score(latest, df)

        return {
            'symbol': symbol, 'timestamp': datetime.now(), 'price':
latest.get('close', 0),
            'volatility': latest.get('volatility', 0.05),
            'volume_anomaly': volume_anomalies[-1] == -1,
            'price_anomaly': price_anomalies[-1] == -1, 'risk_score':
risk_score,
            'social_signal': 1 if symbol in [t[1:] for s in
self.social_signals for t in s['tickers']] else 0
        }

    def calculate_risk_score(self, candle, df):
        """Enhanced risk scoring with multiple factors"""
        if df.empty: return 0
        vol_ma = df['volume'].rolling(60).mean().iloc[-1] if len(df)
>= 60 else df['volume'].mean()
        vol_ratio = candle['volume'] / vol_ma if vol_ma > 0 else 1
        price_ma = df['close'].rolling(60).mean().iloc[-1] if len(df)
>= 60 else df['close'].mean()
        price_std = df['close'].std() if len(df) > 1 else 0.1
        price_dev = (candle['close'] - price_ma) / price_std if
price_std > 0 else 0
        volatility = candle.get('volatility', 0.05)
        score = (0.5 * min(vol_ratio, 10) + 0.3 * abs(price_dev) * 10
+ 0.2 * volatility * 100)
        return min(99, max(0, score))

    def run_surveillance(self):
```

```
        """Main surveillance loop"""
        print("🚀 Starting integrated surveillance...")
        while True:
            try:
                # Process social signals from Telegram
                for signal in self.detected_signals:
                    for ticker in signal['tickers']:
                        symbol = ticker[1:]
                        if symbol in self.watchlist:
                            print(f"🔍 Social mention for {symbol}:
{signal['message'][:50]}...")
                self.social_signals.extend(self.detected_signals)
                self.detected_signals.clear()

                # Check market signals
                for symbol in self.watchlist:
                    detection = self.detect_pump_signals(symbol)
                    if detection and detection['risk_score'] > 75:
                        self.handle_alert(detection)

                # Monitor active trades
                self.monitor_active_trades()

                time.sleep(60)
            except KeyboardInterrupt:
                print("🛑 Surveillance stopped by user.")
                self.trader.save_trade_history()
                break
            except Exception as e:
                print(f"⚠️ Unexpected error in surveillance:
{str(e)}")
                time.sleep(60)

    def handle_alert(self, detection):
        """Process alerts and generate signals"""
        print(f"🚨 ALERT: {detection['symbol']} | Risk Score:
{detection['risk_score']:.0f}/100")
        signal = self.trader.generate_signal(detection)
        if signal:
            print(f"📈 Generated trading signal for
{signal['symbol']}:")
            # ... print signal details
            self.execute_trade(signal)

    def execute_trade(self, signal):
        """Execute trade via Binance API"""
        print(f"✅ Executing trade for {signal['symbol']}")
        # Integration with self.trader.execute_trade and actual API
calls would go here
```

```
    def monitor_active_trades(self):
        """Monitor active trades via Binance API"""
        # Integration with self.trader.monitor_trade and API calls
        pass

    async def start_monitoring(self):
        """Start both market and social monitoring"""
        telegram_started = await self.start()
        if telegram_started:
            self.run_surveillance()
        else:
            print("⚠️ Starting surveillance without Telegram
monitoring")
            self.run_surveillance()
        await self.stop()

if __name__ == "__main__":
    # --- CONFIGURATION ---
    # IMPORTANT: Replace with your actual credentials and channel IDs
    BINANCE_API_KEY = os.environ.get('BINANCE_API_KEY',
'your_binance_api_key')
    BINANCE_API_SECRET = os.environ.get('BINANCE_API_SECRET',
'your_binance_api_secret')
    TELEGRAM_API_ID = os.environ.get('TELEGRAM_API_ID',
'your_telegram_api_id')
    TELEGRAM_API_HASH = os.environ.get('TELEGRAM_API_HASH',
'your_telegram_api_hash')

    # Example Telegram channel IDs (these are numeric IDs)
    PUMP_CHANNELS = [123456789, 987654321]

    WATCHLIST = ['BTCUSDT', 'ETHUSDT', 'BNBUSDT', 'SOLUSDT',
'DOGEUSDT']

    detector = AdvancedPumpDetector(
        api_key=BINANCE_API_KEY, api_secret=BINANCE_API_SECRET,
        watchlist=WATCHLIST, telegram_api_id=TELEGRAM_API_ID,
        telegram_api_hash=TELEGRAM_API_HASH,
telegram_channels=PUMP_CHANNELS,
        account_size=10000
    )

    print("--- Crypto Pre-Pump Playbook: Advanced Scanner ---")
    print("--- Monitoring started - Press Ctrl+C to exit ---")

    try:
        asyncio.run(detector.start_monitoring())
    except KeyboardInterrupt:
```

```
    print("\n🛑 System shutdown requested.")
finally:
    print("💾 Saving final state...")
    detector.trader.save_trade_history()
    print("✅ System shutdown complete.")
```

Appendix C: Recommended Tools & Resources

- **Charting & Screening:** TradingView, DexScreener
- **On-Chain Analysis:** Etherscan (for Ethereum & EVM chains), Solscan (for Solana), Bubblemaps (for token holder visualization).
- **News & Sentiment:** X (Twitter), CoinDesk, The Block.
- **Security Tools:** Token Sniffer, GoPlus Security (for contract scanning).