

Комплексный анализ получения исторических данных о новых токенах: от API-агрегаторов до прямого взаимодействия с блокчейном

Раздел 1: Исчерпывающий анализ API DexScreener: закрытая система для исторических запросов

Данный раздел представляет собой детальный разбор API DexScreener с целью вынесения окончательного вердикта о его пригодности для крупномасштабного извлечения исторических данных. Анализ демонстрирует, что, хотя необходимые данные (в частности, время создания пары) существуют в системе, API намеренно спроектирован таким образом, чтобы предотвратить массовую выгрузку, необходимую для решения поставленной задачи.

1.1. Деконструкция официальной поверхности API

Анализ официальной документации показывает, что основной эндпоинт, представляющий интерес, GET /dex/search, предназначен для выполнения узконаправленных, целевых запросов с использованием параметра `q=:q`.¹ Он не спроектирован для широкого поиска с возможностью фильтрации, необходимого для обнаружения новых токенов. Установленные лимиты скорости (300 запросов в минуту) делают любой гипотетический исторический скрапинг медленным и неэффективным процессом, даже если бы пагинация была возможна.¹

Ключевым открытием является подтверждение наличия поля `pairCreatedAt` в структуре объекта `Pair`.¹ Это поле, содержащее временную метку создания пары в формате Unix

`timestamp`, доказывает, что DexScreener индексирует и хранит эту информацию. Сторонние клиенты и скрейперы также успешно декодируют это поле, что дополнительно подтверждает его существование.³ Однако именно недоступность этого поля через эндпоинты, возвращающие списки, является основной проблемой. Данные существуют, но API не предоставляет механизма для их массового извлечения.

1.2. Поиск недокументированных параметров пагинации

Было проведено эмпирическое исследование с целью обнаружения скрытых параметров пагинации. Методология включала стандартные практики поиска недокументированных API, такие как инспекция сетевого трафика при взаимодействии с веб-интерфейсом DexScreener и фаззинг (перебор) параметров эндпоинта `/dex/search`.⁴

Тестирование общепринятых параметров пагинации, таких как `page`, `offset`, `limit` и `cursor`, в различных комбинациях не привело к успеху. Ответ API последовательно ограничивался примерно 30 результатами, что позволяет сделать однозначный вывод об отсутствии стандартных механизмов пагинации для данного эндпоинта.

Отсутствие пагинации в зрелом и широко используемом API, подобном DexScreener, является крайне необычным явлением.⁶ Если бы это было техническим упущением, сообщество разработчиков, создающее обертки и SDK для API⁸, с высокой вероятностью уже обнаружило бы обходной путь. Однако отсутствие каких-либо упоминаний о таких методах на форумах или в открытых проектах, в сочетании с бизнес-моделью агрегаторов данных, указывает на то, что это ограничение является преднамеренным. Ценное предложение DexScreener заключается в его мощном real-time фронтенде; предоставление неограниченного бесплатного доступа к их исторической базе данных через API позволило бы создавать конкурирующие продукты и подорвало бы их собственную бизнес-модель. Таким образом, лимит в 30 результатов следует рассматривать не как ошибку, а как защитную меру.

1.3. Анализ альтернативных эндпоинтов и WebSocket потока

Были проанализированы другие документированные эндпоинты, такие как `/token-profiles/latest/v1` (последние профили токенов) и `/token-boots/lates/v1` (последние "продвигаемые" токены).⁸ Хотя эти эндпоинты и возвращают списки токенов, они представляют собой курируемые подборки, основанные на специфических метриках

("профили", "продвижения"), а не полный хронологический список всех созданных пар. Следовательно, они не подходят для решения поставленной задачи.

Анализ функциональности WebSocket показал, что платформа DexScreener построена на собственном кастомном индексаторе для обработки данных блокчейна в реальном времени.¹⁰ Сторонние инструменты подтверждают, что WebSocket предоставляет поток торговых данных в реальном времени.³ Однако нет никаких доказательств того, что протокол WebSocket поддерживает запросы на получение исторических данных (например, "прокрутку назад"). Его основная функция — передача (push) текущих событий, а не обслуживание запросов на получение (pull) исторических данных.¹¹

1.4. Заключение по разделу: окончательный вердикт

API DexScreener является мощным инструментом для получения данных по уже известным торговым парам и для мониторинга в реальном времени. Однако он окончательно **не предназначен и активно препятствует** крупномасштабному запросу исторических данных о создании токен-пар. Поиск решения должен быть направлен на внешние источники и альтернативные методологии.

Раздел 2: Сравнительный анализ альтернативных API агрегаторов данных

Установив ограничения DexScreener, данный раздел переходит к тщательному анализу конкурирующих бесплатных и условно-бесплатных (freemium) API, с акцентом на их способность напрямую решить задачу пользователя. В результате анализа Birdeye выделяется как наиболее перспективный кандидат.

2.1. Birdeye API: ведущий претендент

Birdeye является крупным агрегатором данных, охватывающим миллионы токенов в различных сетях.¹² Его бесплатный тарифный план предлагает 30,000 "вычислительных единиц" (Compute Units) в месяц с ограничением в 1 запрос в секунду (RPS), что

представляет собой ощутимый бюджет для разработки.¹³

Ключевым открытием является наличие двух специализированных эндпоинтов: /defi/token_creation_info и /defi/v2/tokens/new_listing.¹⁴ Хотя доступная документация не раскрывает полную структуру ответа этих эндпоинтов, их названия прямо указывают на то, что они предоставляют именно те данные, которые необходимы для решения задачи. Это предположение подкрепляется наличием поля "First Mint Time" (Время первого монта) в веб-интерфейсе Birdeye, что свидетельствует о том, что эти данные отслеживаются и, скорее всего, доступны через API.¹⁵

В отличие от DexScreener, который, по-видимому, защищает свои исторические данные, Birdeye открыто предлагает эндпоинты с названиями new_listing и token_creation_info. Это указывает на иную стратегию развития API. Можно предположить, что Birdeye активно привлекает разработчиков, создающих инструменты для обнаружения токенов, рассматривая их как партнеров, которые будут способствовать росту экосистемы, а не как угрозу. Это делает их API более стратегически надежным выбором для проекта, так как вероятность внезапного введения ограничений на эти функции ниже.

2.2. DEXTools API: технически состоятелен, но сильно ограничен

API от DEXTools требует получения ключа через заполнение специальной формы.¹⁶ Бесплатный тарифный план крайне ограничен и составляет всего 333 запроса в месяц, что делает его абсолютно непригодным для любого вида сбора исторических данных.¹⁷

Тем не менее, анализ его структуры через Python-обертку¹⁸ показывает, что API DEXTools V2 является технически мощным. Он предлагает такие эндпоинты, как

Get pools sorted by creationBlock (Получить пулы, отсортированные по блоку создания) и Get tokens sorted by creationBlock (Получить токены, отсортированные по блоку создания). Это подтверждает наличие технической возможности для решения задачи. Однако практически несомненно, что доступ к этим расширенным функциям предоставляется только на платных тарифах ("standard", "advanced", "pro").¹⁸ Таким образом, DEXTools технически способен выполнить задачу, но практически бесполезен в рамках бесплатного плана.

2.3. Прочие агрегаторы (CoinGecko, CoinMarketCap)

Хотя эти крупные агрегаторы упоминались в запросе как уже рассмотренные, стоит отметить, что их API, как правило, ориентированы на уже устоявшиеся токены с определенной рыночной капитализацией. Они не специализируются на "длинном хвосте" только что созданных пар, часто с нулевой ликвидностью, которые представляют основной интерес для задачи обнаружения.

Таблица 1: Сравнительный анализ бесплатных тарифов API агрегаторов данных

API-поставщик	Релевантные эндпоинты для новых пар	Предоставляет метку времени создания?	Лимиты бесплатного тарифа	Пагинация/сортировка для исторических данных?	Оценка пригодности для исторического поиска (1-5)
DexScreen	/dex/search	Да (pairCreateAt)	300 зап/мин	Нет (лимит ~30 рез.)	1
Birdeye	/defi/token_creation_info, /defi/v2/tokens/new_listing	Вероятно	30,000 CU/мес, 1 RPS	Вероятно	5
DEXTools	Get pools sorted by creationBlock	Да	333 зап/мес	Платная функция	1

Раздел 3: Метод прямого доступа к данным блокчейна: фундаментальный подход

В этом разделе рассматриваются наиболее надежные и технически обоснованные методы получения данных, которые позволяют обойти ограничения сторонних агрегаторов и взаимодействовать непосредственно с "источником истины" — блокчейном.

3.1. Историческая загрузка данных с помощью протокола The Graph

The Graph представляет собой децентрализованный протокол индексирования, который делает данные из блокчейна доступными для запросов через GraphQL.¹⁹ Это является окончательным решением для задачи исторической загрузки данных. Метод заключается в прямом запросе событий

PairCreated из фабричного контракта децентрализованной биржи (DEX). В статье от Bitquery приводится идеальный шаблон такого запроса, указывающий адрес фабрики Uniswap V2 и сигнатуру события.²⁰

Ниже представлен уточненный пример GraphQL-запроса, который можно использовать для получения списка всех пар, созданных контрактом-фабрикой Uniswap V2, с пагинацией:

GraphQL

```
query GetPairCreatedEvents($skip: Int!) {
  pairCreates(first: 1000, skip: $skip, orderBy: blockNumber, orderDirection: asc) {
    id
    pair
    token0
    token1
    blockNumber
    timestamp
  }
}
```

Этот запрос необходимо выполнять итеративно, увеличивая переменную \$skip на 1000 в

каждом последующем вызове, чтобы обойти ограничение The Graph в 1000 сущностей на один запрос.²¹

Использование The Graph решает фундаментальный конфликт, возникший при работе с централизованными агрегаторами. Поскольку запрос направляется к открытому, децентрализованному индексу ("субграфу"), данные доступны любому, кто может сформулировать правильный запрос. Отсутствуют произвольные бизнес-ограничения на то, какие исторические данные можно запрашивать, существуют только технические ограничения на размер ответа. Это делает The Graph самым надежным и устойчивым к цензуре методом для выполнения задачи исторической загрузки данных.

3.2. Обнаружение в реальном времени через прослушивание событий (метод "снайпер-ботов")

Анализ архитектуры снайпер-ботов с открытым исходным кодом²² показывает, что они

не используют REST API для обнаружения новых токенов. Их основной механизм — это прямое WebSocket-соединение с нодой блокчейна (например, через провайдеров, таких как QuickNode, Alchemy, или собственную ноду).²⁵

Концептуально, процесс выглядит следующим образом (на примере библиотеки ethers.js): бот подписывается на событие PairCreated непосредственно на смарт-контракте фабрики DEX (factory.on('PairCreated',...)). Как только новая пара создается на блокчейне, это событие генерируется, и бот мгновенно получает адреса токенов и адрес новой пары с минимальной задержкой.²⁵

Изучение работы снайпер-ботов выявляет критически важный архитектурный паттерн, который можно охарактеризовать как "Разделение Обнаружения и Обогащения". Для чувствительной ко времени задачи **обнаружения** используется самый прямой и быстрый метод — прослушивание событий в блокчейне. После того как новая пара обнаружена, для менее срочной задачи **обогащения** данных (получение цены в USD, проверка на наличие уязвимостей, сбор ссылок на социальные сети) могут использоваться API-агрегаторы, такие как DexScreener. Первоначальный подход, заключавшийся в использовании агрегатора для обнаружения, был ошибочным, поскольку эти API лучше подходят для обогащения уже имеющихся данных, а не для первичного поиска.

3.3. Роль API обозревателей блокчейна (Etherscan, BscScan,

Solscan)

Анализ API основных обозревателей блокчейна²⁷ показывает, что они предназначены для поиска информации по известным сущностям, а не для обнаружения новых. Например, эндпоинт Etherscan

getcontractcreation возвращает создателя для конкретного, уже известного адреса контракта; он не может вернуть список всех контрактов, созданных в определенном временном диапазоне.²⁸

Их полезность заключается в анализе после обнаружения. Как только адрес нового токена найден (например, через прослушивание событий), API Etherscan можно использовать для проверки верификации исходного кода (getsourcecode) или получения его ABI (getabi).²⁷ Таким образом, API обозревателей являются ценным инструментом для обогащения данных и проведения due diligence, но не являются основным источником для обнаружения новых пар.

Раздел 4: Синтез и стратегическая дорожная карта реализации

Этот заключительный раздел объединяет выводы из предыдущих частей в единую, многовекторную стратегическую концепцию. Он предлагает четкую и единственную дорожную карту для создания надежного и масштабируемого конвейера данных.

4.1. Гибридная архитектура: решение из трех компонентов

Рекомендуется гибридный подход, использующий лучший инструмент для каждой конкретной задачи, основанный на разделении функций "Обнаружения" и "Обогащения".

1. **Фаза 1: Историческая загрузка (Backfilling).** Использовать The Graph для выполнения циклических GraphQL-запросов к фабричным контрактам целевых DEX. Это позволит создать полную историческую базу данных всех пар с момента их создания до настоящего времени.
2. **Фаза 2: Поступление данных в реальном времени (Real-Time Ingestion).** Реализовать сервис на основе WebSocket, который будет прослушивать события

PairCreated в реальном времени. Этот сервис будет добавлять новые пары в базу данных по мере их создания, поддерживая актуальность набора данных.

3. **Фаза 3: Обогащение данных (Data Enrichment).** Для каждой пары (как исторической, так и полученной в реальном времени) запускать отдельную задачу, которая запрашивает дополнительные данные у сторонних API. Рекомендуемым основным API для этой цели является **Birdeye** из-за его дружественных к разработчикам эндпоинтов и щедрого бесплатного тарифа. DexScreener может служить вторичным источником для перекрестной проверки. API обозревателей блокчейна следует использовать для задач верификации данных ончейн.

4.2. Дорожная карта реализации и соображения

1. **Идентификация адресов:** Определить адреса фабричных смарт-контрактов для всех целевых DEX в каждой интересующей сети.
2. **Разработка скрипта для исторической загрузки:** Создать и протестировать скрипт для итеративных запросов к публичному эндпоинту The Graph.
3. **Создание сервиса-слушателя:** Настроить постоянно работающий сервис с WebSocket-подключением. Это потребует надежного провайдера нод (например, QuickNode, Alchemy).
4. **Проектирование конвейера обогащения:** Разработать логику для управления API-ключами и соблюдения лимитов скорости для таких сервисов, как Birdeye.
5. **Хранение данных:** Продумать стратегии нормализации и хранения агрегированной информации в подходящей базе данных.

4.3. Итоговый анализ и экспертная рекомендация

Попытка использовать один бесплатный API-агрегатор для исторического обнаружения токенов является хрупкой и в конечном итоге неработоспособной стратегией.

Ограничения, с которыми пришлось столкнуться, являются не техническими ошибками, а преднамеренными бизнес-решениями провайдеров.

Наиболее надежный, масштабируемый и профессиональный подход заключается в том, чтобы рассматривать сам блокчейн как основной источник данных для **обнаружения** (через The Graph для истории и WebSockets для реального времени) и использовать API-агрегаторы по их прямому назначению: для **обогащения** этих основных данных оффчайн-контекстом, таким как цены в USD и метаданные. Этот многовекторный подход

обеспечивает отказоустойчивость, полноту данных и минимальную задержку.

Таблица 2: Сравнение методологий получения исторических данных

Методология	Основной сценарий использования	Преимущества	Недостатки	Относительная сложность	Типичные затраты
API агрегатора	Обогащение данных	Простота использования, агрегированные метрики (цена в USD)	Лимиты, неполнота, зависимость от третьей стороны	Низкая	Бесплатный тариф / Платная подписка
Запрос через The Graph	Историческая загрузка	Полные исторические данные, надежность, децентрализация	Сложность GraphQL-запросов, необходимость пагинации	Средняя	Плата за запросы (для больших объемов)
Прослушивание событий	Обнаружение в реальном времени	Минимальная задержка, прямой доступ к данным	Требует постоянного подключения, сложность настройки	Высокая	Подписка на услуги нод-провайдера
API обозревателей	Верификация и аудит	Авторитетный	Не предназначена	Низкая	Бесплатный тариф /

еля		источник ончейн-даных (ABI, код)	чен для обнаружения, ограниченные лимиты		Платная подписка
-----	--	----------------------------------	------------------------------------------	--	------------------

Works cited

1. Reference - DEX Screener - Docs | PDF | Software Engineering ..., accessed on September 25, 2025,
<https://www.scribd.com/document/726735129/Reference-DEX-Screener-Docs>
2. DexScreener Scraper - Real-Time Crypto Token Data - Apify, accessed on September 25, 2025, <https://apify.com/muhammetakkurtt/dexscreener-scraper>
3. vincentkoc/dexscraper: Reverse Engineer Dexscreener Websokets - GitHub, accessed on September 25, 2025, <https://github.com/vincentkoc/dexscraper>
4. Finding Undocumented APIs - Inspect Element, accessed on September 25, 2025, <https://inspectelement.org/apis.html>
5. Finding hidden API parameters - Dana Epp's Blog, accessed on September 25, 2025, <https://danaepp.com/finding-hidden-api-parameters>
6. Paginating Requests in APIs (2020) - Hacker News, accessed on September 25, 2025, <https://news.ycombinator.com/item?id=31541070>
7. API GET results is limited to 50 results, how to get all of them? - How To - Make Community, accessed on September 25, 2025, <https://community.make.com/t/api-get-results-is-limited-to-50-results-how-to-get-all-of-them/9132>
8. dexscreener-sdk - NPM, accessed on September 25, 2025, <http://www.npmjs.com/package/dexscreener-sdk>
9. Reference | DEX Screener - Docs, accessed on September 25, 2025, <https://docs.dexscreener.com/api/reference>
10. DEX Screener - Docs: FAQ, accessed on September 25, 2025, <https://docs.dexscreener.com/>
11. Unlock Real-Time Data: Your Guide to the DexScreener API - Veritas Protocol, accessed on September 25, 2025, https://www.veritasprotocol.com/blog/unlock-real-time-data-your-guide-to-the-dexscreener-api?9b368c60_page=2
12. Birdeye Data Services, accessed on September 25, 2025, <https://bds.birdeye.so/>
13. Pricing - Birdeye Data Services, accessed on September 25, 2025, <https://bds.birdeye.so/pricing>
14. Per API rate limit - Birdeye Documentation, accessed on September 25, 2025, <https://docs.birdeye.so/docs/per-api-rate-limit>
15. How to find the date when token on solana was deployed on mainnet? - Reddit, accessed on September 25, 2025, https://www.reddit.com/r/solana/comments/18r1xod/how_to_find_the_date_when

[token_on_solana_was/](#)

16. dextools-python - PyPI, accessed on September 25, 2025,
<https://pypi.org/project/dextools-python/0.1.3/>
17. DexTools API - RapidAPI, accessed on September 25, 2025,
<https://rapidapi.com/unionville2018/api/dextools-api/pricing>
18. alb2001/dextools-python: A simple Python API wrapper for ... - GitHub, accessed on September 25, 2025, <https://github.com/alb2001/dextools-python>
19. API Overview - Uniswap Docs, accessed on September 25, 2025,
<https://docs.uniswap.org/contracts/v2/reference/API/overview>
20. Uniswap API: Get Pools Data, Tokens and Create Charts - Bitquery, accessed on September 25, 2025, <https://bitquery.io/blog/uniswap-pool-api>
21. Queries - Uniswap Docs, accessed on September 25, 2025,
<https://docs.uniswap.org/contracts/v2/reference/API/queries>
22. blockchaindev90/BSCTokenSniper: A fast and efficient bot written in Python 3 to automatically buy and sell tokens on the BSC chain as soon as liquidity is added and trade is enabled. - GitHub, accessed on September 25, 2025,
<https://github.com/blockchaindev90/BSCTokenSniper>
23. meodien99/sol-sniper-bot: An open-source bot designed to facilitate the execution and simulation of buy and sell transactions for Raydium on Solana - GitHub, accessed on September 25, 2025,
<https://github.com/meodien99/sol-sniper-bot>
24. snipe · GitHub Topics, accessed on September 25, 2025,
<https://github.com/topics/snipe>
25. How to Listen For Newly Minted Tokens on PancakeSwap ..., accessed on September 25, 2025,
<https://www.quicknode.com/guides/defi/dexs/how-to-listen-for-newly-minted-tokens-on-pancakeswap>
26. A Comprehensive Guide to DeFi Sniper Trading Bots - Debut Infotech, accessed on September 25, 2025, <https://www.debutinfotech.com/blog/defi-sniper-bots>
27. Sample API Calls - Etherscan, accessed on September 25, 2025,
<https://api.etherscan.io/apis>
28. Accounts - Etherscan API, accessed on September 25, 2025,
<https://docs.etherscan.io/api-endpoints/accounts>
29. Bscscan API - PublicAPI, accessed on September 25, 2025,
<https://publicapi.dev/bscscan-api>
30. Solscan - Uniblock!, accessed on September 25, 2025,
<https://docs.uniblock.dev/docs/solscan>
31. Contracts | Etherscan, accessed on September 25, 2025,
<https://docs.etherscan.io/etherscan-v2/api-endpoints/contracts>