www.cyberok.ru
info@cyberok.ru
+7 (495) 137-7337

Что нам стоит хост пофаззить: методы оптимизации фаззинга веб-директорий и файлов

Станислав Савченко – ведущий специалист базы знаний, СайберОК

Аннотация

Фаззинг — один из ключевых методов поиска уязвимостей и скрытых ресурсов на веб-серверах. Однако классический подход, использующий статические словари, часто неэффективен: он генерирует огромное количество бесполезных запросов и при этом может пропускать специфичные для целевого приложения директории и файлы. В данной статье рассматриваются методы оптимизации этого процесса с использованием краулинга, детектирования технологий и больших языковых моделей (LLM). Мы покажем, как комбинация этих методов позволяет значительно расширить охват поиска при одновременном сокращении количества запросов к серверу, делая процесс анализа более целенаправленным и эффективным.

Введение

В рамках нашей услуги PentOps – непрерывный автоматизированный пентест – мы постоянно совершенствуем методы и инструменты наступательной безопасности. Наша цель - сделать их точнее, быстрее и эффективнее, чтобы находить уязвимости раньше, чем их найдут злоумышленники. Сегодня я расскажу об одном из недавних улучшений, которое помогло нам приблизиться к этой цели.

Поиск скрытых директорий и файлов — одна из рутинных, но критически важных задач при оценке безопасности веб-приложений. Стандартный подход, однако, часто напоминает

поиск иголки в стоге сена. Инструменты вроде ffuf перебирают тысячи путей из заранее подготовленного универсального словаря, который содержит всё — от путей для PHP и ASPX до файлов бэкапов и конфигураций для десятков фреймворков. При тестировании же конкретного хоста (например, сайта на Drupal) это приводит к тому, что подавляющее большинство запросов оказывается заведомо нерелевантными, создавая избыточную нагрузку и повышая шанс пропустить реально существующие уязвимости. Таким образом, мы сталкиваемся с двумя задачами оптимизации:

1. **Расширение охвата:** найти больше скрытых ресурсов, чем позволяет стандартный словарь, особенно если ресурсы кастомизированы.
2. **Сокращение запросов:** уменьшить количество HTTP-запросов к серверу, убрав из словаря заведомо неподходящие пути, что ускоряет анализ и снижает нагрузку на сервер.

В качестве тестового стенда для демонстрации методов использовался контейнер с веб-приложением на базе Drupal / PHP / Apache.

Как расширить охват и найти больше скрытых ресурсов

Отправная точка: Классический фаззинг

Используя стандартный общедоступный словарь объемом 9057 записей и запустив ffuf, на тестовом стенде было обнаружено **16** существующих эндпоинтов. Это наш базовый уровень, который мы будем стараться улучшить.

Параметр	Количество запросов	Обнаружено эндпоинтов
Static dict	9057	16

Динамический словарь на основе краулинга и LLM

Первый шаг к оптимизации — создание динамического словаря, зависящего от контекста конкретного хоста. Отличный пример такого подхода — утилита **Brainstorm** (от Invicti Security). Ее логика работы заключается в следующем:

1. **Краулинг:** сначала инструмент собирает все доступные ссылки с главной страницы сайта (по умолчанию – 25 ссылок).
2. **Генерация гипотез:** Собранные URL передаются большой языковой модели (LLM) с задачей: «На основе этих путей предложи еще 50 возможных эндпоинтов».
3. **Верификация:** Сгенерированные пути отправляются в ffuf для фаззинга.
4. **Итерация:** Найденные рабочие эндпоинты снова передаются в LLM для генерации новых гипотез. Этот цикл повторяется несколько десятков раз (например – 20).

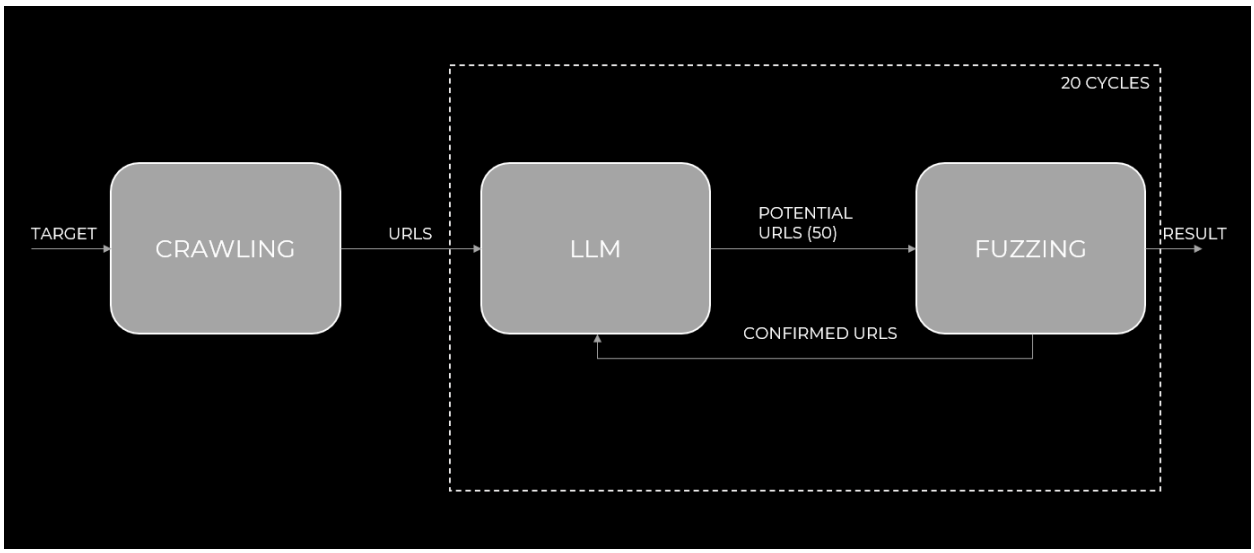

Рис.1 Автор: С. О. Савченко, 2025.

В результате утилита возвращает некоторое количество эндпоинтов, которые точно есть на хосте, и которые зависят от того, что на нем было обнаружено ранее.

Запустив похожую утилиту, но выполняющую на этапе подготовки рекурсивный поиск ссылок по всей глубине хоста, этот метод позволил обнаружить **21** эндпоинт. Количество запросов к хосту составило всего 1000 запросов (50 URL в каждой из 20 итераций), но это уже целевой, «умный» фаззинг. Помимо этого, было совершено 20 запросов к LLM.

Параметр	Количество запросов	Обнаружено эндпоинтов
Static dict	9057	16
Crawling + LLM	< 1000 + 20 LLM	21

Следует отметить, что 1000 запросов – это верхняя оценка, так как LLM зачастую предлагает повторяющиеся URL, и один раз пометив URL как несуществующий, можно не совершать повторные его вызовы, сократив тем самым нагрузку на хост и уменьшив время работы.

Учет технологического стека

Метод Brainstorm можно улучшить, если дать LLM больше информации о цели. Эту информацию легко извлечь, например, из заголовков ответа хоста:

```
"cache_control: must-revalidate, no-cache, private",
"content_language: ru",
"content_type: text/html; charset=UTF-8",
"date: Mon, 15 Sep 2025 12:37:33 GMT",
"expires: Sun, 19 Nov 1978 05:00:00 GMT",
"server: Apache/2.4.65 (Debian)",
"vary: Accept-Encoding",
"x_content_type_options: nosniff",
"x_drupal_cache: HIT",
"x_drupal_dynamic_cache: MISS",
"x_frame_options: SAMEORIGIN",
"x_generator: Drupal 10 (https://www.drupal.org)",
"x_powered_by: PHP/8.4.12"
```

Рис.2 Автор: С. О. Савченко, 2025.

Уже на этом этапе можно сказать, что на хосте используется Drupal, Apache и PHP, а следовательно, обнаружить на нем пути характерные для ASP.NET или оборудования Cisco маловероятно.

Но если начинать анализировать используемые технологии, то лучше использовать специализированные утилиты:

- **httpx** от Project Discovery, который под капотом имеет модифицированный Wappalyzer и позволяет детектировать множество технологий, исходя из тела и заголовков ответа главной страницы;
- **nuclei** от Project Discovery, который, будучи запущен с тегом **tech**, использует шаблоны для детекта технологий из специализированных URL.

Кроме того, можно в LLM передать заголовки и тело ответа хоста, чтобы была возможность задетектировать технологии, которые оказались незамеченными двумя предыдущими утилитами. Итоговая схема выглядит следующим образом:

Рис 3. Автор: С. О. Савченко, 2025.

Как можно увидеть, технологии были определены примерно одинаковые, но LLM дополнительно из тела ответа смог вытащить название используемой темы – Olivero Theme – что потенциально может расширить пространство потенциальных эндпоинтов.

Помимо указанных httpx и nuclei, можно использовать любые другие утилиты для детектирования используемых технологий, в том числе Aquatone и EyeWitness, которые с помощью headless браузера обращаются к хосту, делают скриншот главной страницы и передают его LLM с целью визуального опознания технологий.

Далее можно использовать схему Brainstorm, но отправить в качестве входных данных не результаты краулинга, а результаты техдетекта,

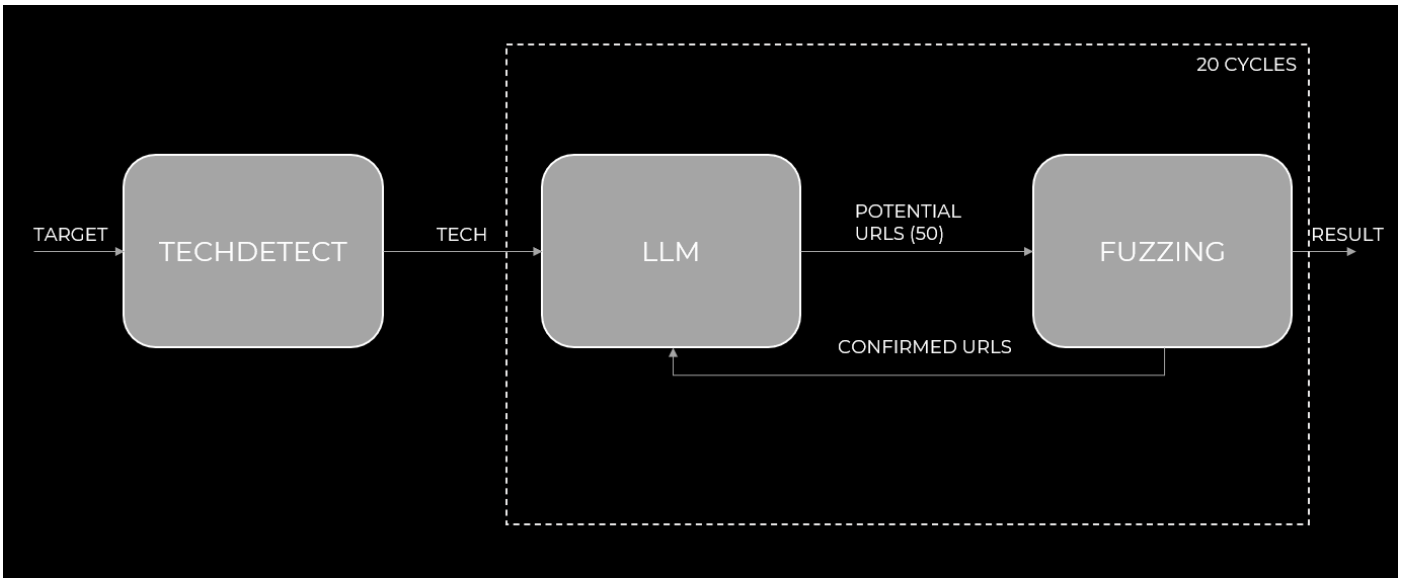

Рис.4 Автор: С. О. Савченко, 2025.

Тестирование такой схемы на стенде покажет не совсем обнадеживающие результаты - при таком же количестве запросов к хосту и LLM, данный метод позволил обнаружить всего 18 эндпоинтов:

Параметр	Количество запросов	Обнаружено эндпоинтов
Static dict	9057	16
Crawling + LLM	< 1000 + 20 LLM	21
Techdetect + LLM	< 1000 + 20 LLM	18

Однако остается возможность совместить эти два метода, направив на вход LLM всё что есть – обнаруженные технологии и результаты краулинга:

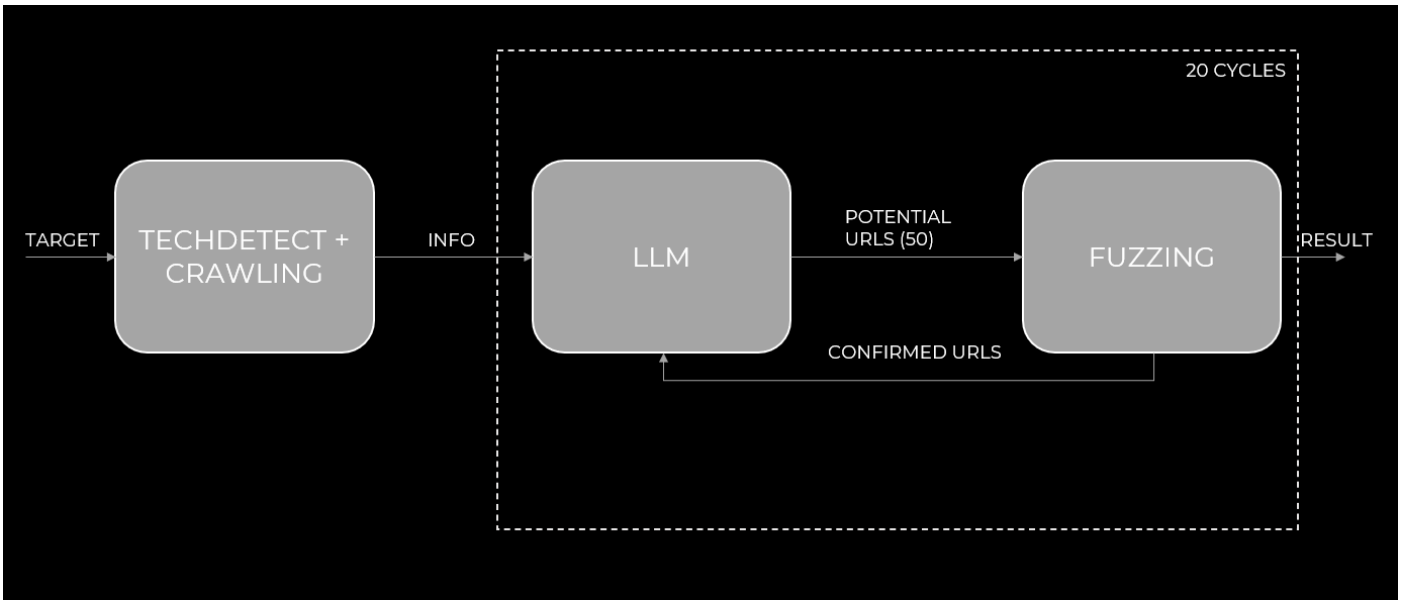

Рис.5 Автор: С. О. Савченко, 2025.

При таком подходе количество обнаруженных эндпоинтов значительно возрастает, при том что количество отправленных запросов остается тем же – не более 1000 к хосту и 20 к LLM:

Параметр	Количество запросов	Обнаружено эндпоинтов
Static Dict	9057	16
Crawling + LLM	< 1000 + 20 LLM	21
Techdetect + LLM	< 1000 + 20 LLM	18
Crawling + Techdetect + LLM	< 1000 + 20 LLM	33

Соединив же динамический словарь, созданный специального для исследуемого хоста, и статический словарь, можно достичь максимального результата, но ценой большого количества запросов:

Параметр	Количество запросов	Обнаружено эндпоинтов
Static Dict	9057	16
Crawling + LLM	< 1000 + 20 LLM	21
Techdetect + LLM	< 1000 + 20 LLM	18
Crawling + Techdetect + LLM	< 1000 + 20 LLM	33
Static Dict + Crawling + Techdetect + LLM	<10057 + 20 LLM	42

Как сократить количество запросов без потери качества

Что делать, если мы не можем позволить себе десятки тысяч запросов на один хост? Нужно сделать статический словарь «умнее».

Контекстное усечение словаря (Crop Dict)

www.cyberok.ru
info@cyberok.ru
+7 (495) 137-7337

Идея проста: как мы помним, наш стандартный словарь содержит множество эндпоинтов, которые могут быть нерелевантны для отдельно взятого хоста. Поэтому можно передать LLM наш словарь и список технологий хоста с просьбой убрать всё, что не соответствует контексту.

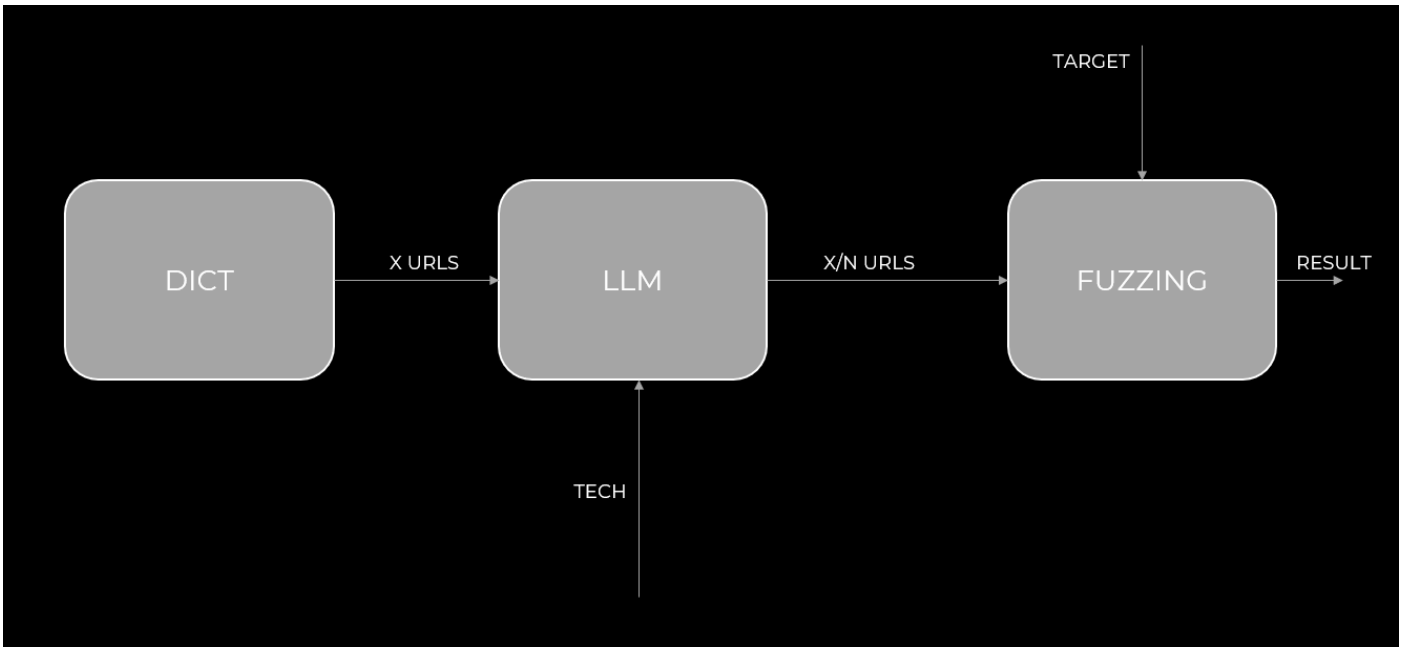

Рис. 6 Автор: С. О. Савченко, 2025.

Но здесь возникает проблема – даже в самом простом словаре из 9000 записей содержится порядка 157 тысяч символов, что превышает лимит контекста большинства LLM. Решением данной проблемы может быть разбиение словаря на **чанки** по 100-200 записей. Каждый чанк поочередно отправляется в LLM для фильтрации. После обработки всех чанков получается усеченный словарь.

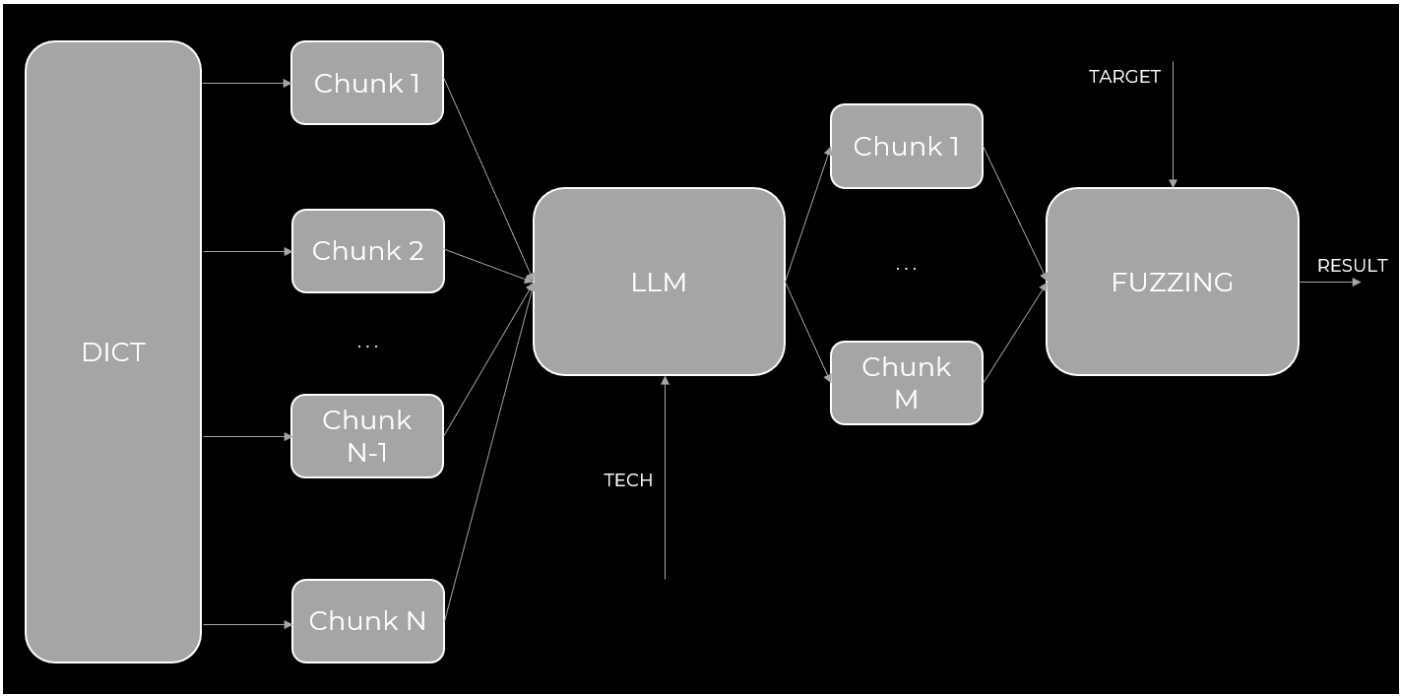

Рис.7 Автор: С. О. Савченко, 2025.

Опытным путем было выявлено, что оптимальный размер чанка составляет 100-200 записей. Если чанк будет слишком большим, есть опасность не влезть в размер окна контекста или получить галлюцинации LLM. При слишком маленьком размере чанка значительно увеличивается количество запросов к LLM, что приводит к удорожанию процесса, особенно на больших проектах.

Использовав 100 чанков и сравнив словари, можем заметить, что усеченный словарь значительно “похудел” – почти в три раза:

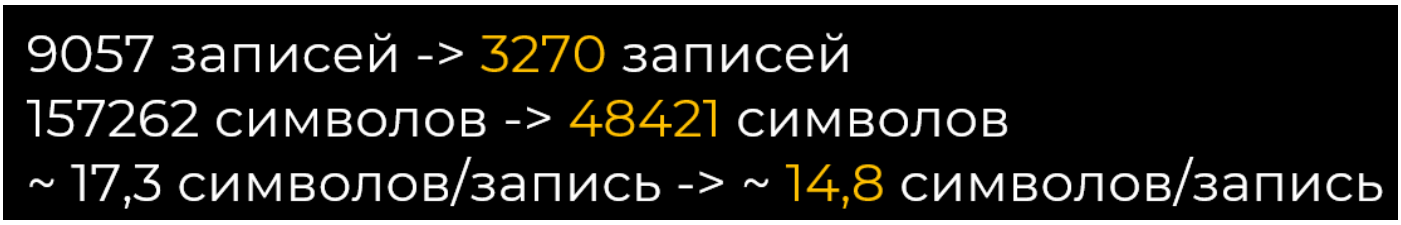

Рис.8 Автор: С. О. Савченко, 2025.

Использовав усеченный словарь вместе с ffuf, добьемся уменьшения количества запросов в три раза ценой потери всего одного эндпоинта по сравнению с статичным словарем

Параметр	Количество запросов	Обнаружено эндпоинтов
Static Dict	9057	16
Crawling + LLM	< 1000 + 20 LLM	21
Techdetect + LLM	< 1000 + 20 LLM	18
Crawling + Techdetect + LLM	< 1000 + 20 LLM	33
Static dict + Crawling + Techdetect + LLM	<10057 + 20 LLM	42
Crop dict	3270 + 100 LLM	15

Но в таблице можно также увидеть и главную проблему данного подхода – для контекстного усечения словаря потребовалось 100 обращений к LLM, что приводит к плохой масштабируемости процесса на большом скоупе.

Предварительно размеченный словарь (Pre-marked Dict)

Чтобы избежать многократной дорогостоящей фильтрации словаря для каждого хоста, можно заранее подготовить такой словарь и переиспользовать при необходимости.

Идея проста: взяв список всех возможных детектируемых технологий (например, из шаблона nuclei tech-detect) и сопоставив с стандартным словарем, можно каждой технологии в соответствие поставить перечень эндпоинтов, которые наиболее характерны для нее. Например:

```
"apache": [
    "server-status",
    "server-info",
    ".htaccess"
],
"php": [
    "index.php",
    "info.php",
    "login.php"
],
"wordpress": [
    "wp-login.php",
    "wp-admin/",
    "wp-content/"
]
```

Рис. 9 Автор: С. О. Савченко, 2025.

Так как процедура довольно трудоемкая, целесообразнее будет выполнить ее с помощью LLM, по аналогии с предыдущим методом:

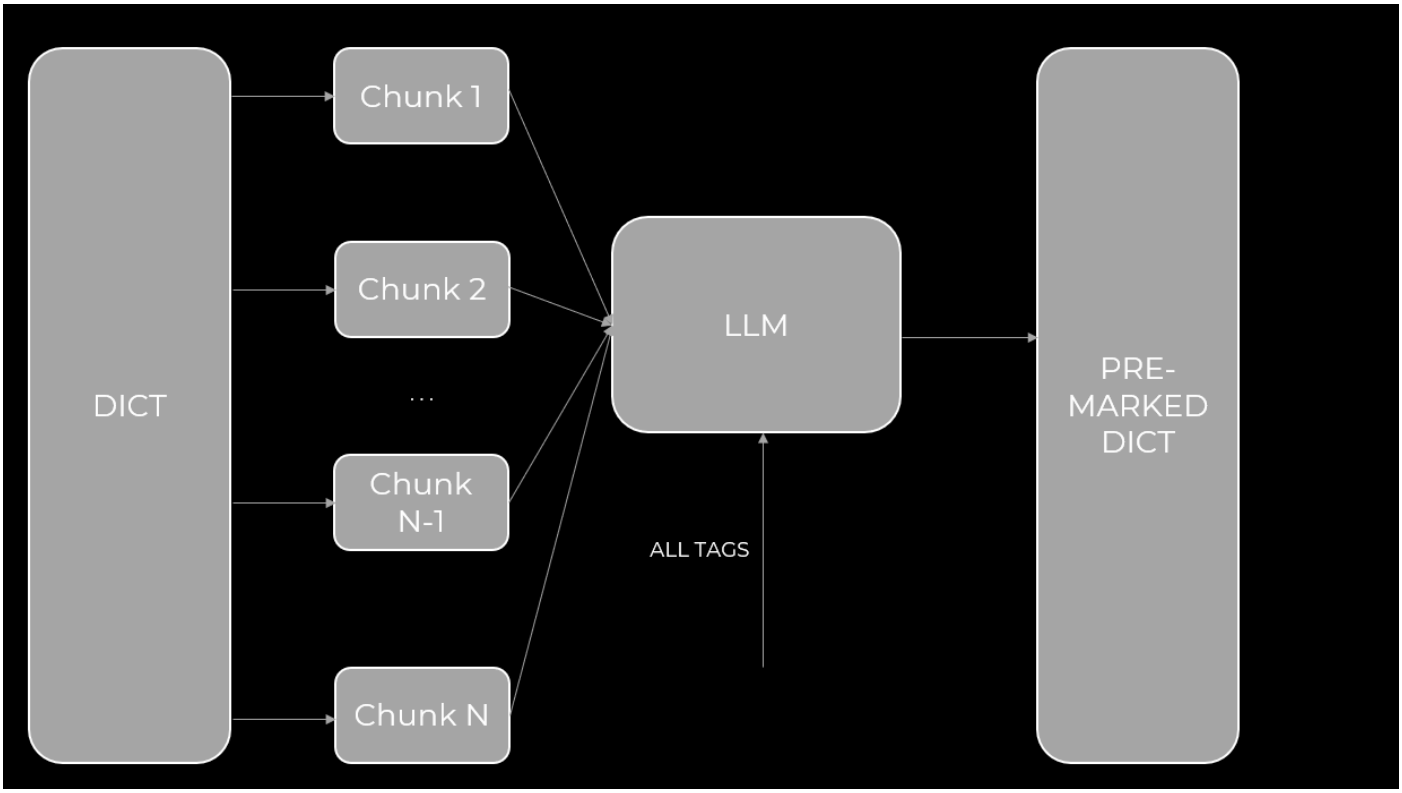

Рис.10 Автор: С. О. Савченко, 2025.

Далее, имея размеченный словарь, можно для каждого хоста собирать технологии, как указывалось в начале статьи, передавать в LLM вместе со списком всех возможных технологий и просить выбрать наиболее подходящие. После этого по имеющимся технологиям можно

выбрать из размеченного словаря только те записи, которые действительно могут быть полезны:

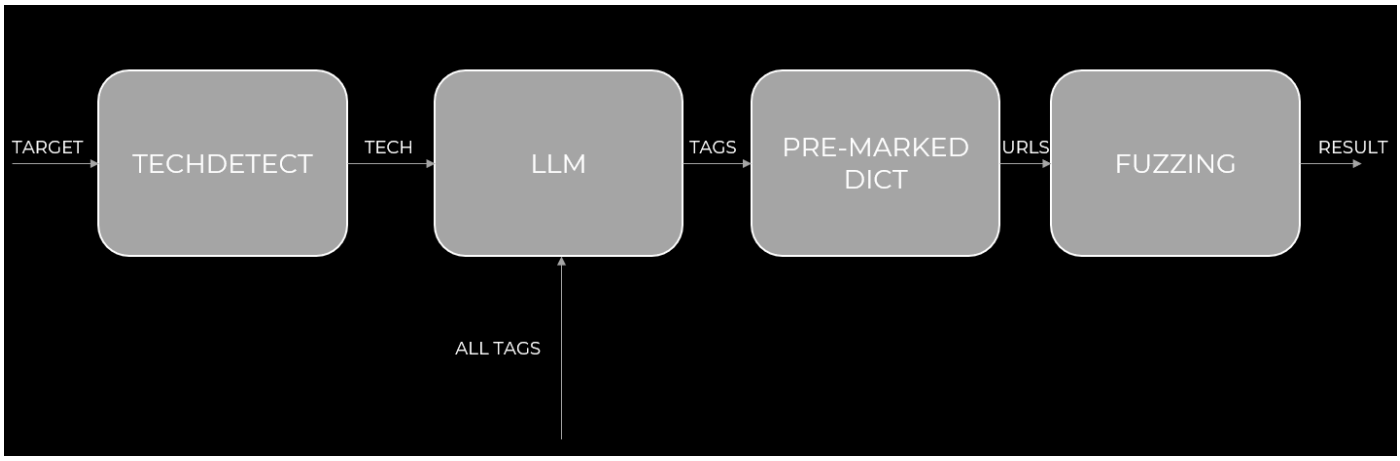

Рис.11 Автор: С. О. Савченко, 2025.

Реализовав весь процесс и запустив ffuf с полученным списком эндпоинтов, были получены следующие результаты:

Параметр	Количество запросов	Обнаружено эндпоинтов
Static dict	9057	16
Crawling + LLM	< 1000 + 20 LLM	21
Techdetect + LLM	< 1000 + 20 LLM	18
Crawling + Techdetect + LLM	< 1000 + 20 LLM	33
Static dict + Crawling + Techdetect + LLM	<10057 + 20 LLM	42
Crop dict	3270 + 100 LLM	15
Pre-marked dict	1302 (100 LLM)	12

Отправив всего 1302 запроса к хосту, смогли обнаружить 12 эндпоинтов. При этом было выполнено всего 100 обращений к LLM, после чего словарь можно переиспользовать сколько угодно раз.

Используя “best practice” и совмещая с динамическим словарем описанные методы сокращения словаря (усеченный словарь и предварительно размеченный), можно добиться гораздо лучших результатов:

Параметр	Количество запросов	Обнаружено эндпоинтов
Static dict	9057	16
Crawling + LLM	< 1000 + 20 LLM	21
Techdetect + LLM	< 1000 + 20 LLM	18
Crawling + Techdetect + LLM	< 1000 + 20 LLM	33
Static dict + Crawling + Techdetect + LLM	<10057 + 20 LLM	42
Crop dict	3270 + 100 LLM	15
Pre-marked dict	1302 (100 LLM)	12
Crop dict + Crawling + Techdetect + LLM	< 4270 + 120 LLM	40
Pre-marked dict + Crawling + Techdetect + LLM	< 2302 + 20 LLM (100 LLM)	39

- **Комбинация Усеченный словарь + Динамический словарь:** 40 эндпоинтов, ~4270 запросов, 120 запросов к LLM на каждый хост.
- **Комбинация Предварительно размеченный словарь + Динамический словарь:** 39 эндпоинтов, ~2000 запросов, всего 20 запросов к LLM для этого хоста (плюс 100 однократных запросов на разметку).

Выводы

Таким образом, можно сделать следующие выводы:

- Для точечного, глубокого анализа одного критически важного хоста, если важен каждый скрытый эндпоинт или нет ограничений на количество отправляемых запросов,

лучше использовать большой статический словарь в сочетании с динамическим словарем (Crawling + Techdetect + LLM).

- Если количество запросов всё же важно, то следует отталкиваться от того, какую погрешность мы готовы принять и сколько хостов у нас в скоупе. При одном хосте разумнее использовать усеченный словарь, при большом скоупе – предварительно размеченный, в любом случае дополнив их динамическим словарем.

Заключение и перспективы

Представленные методы показывают, как интеграция больших языковых моделей в традиционные инструменты пентеста позволяет перейти от «грубой силы» к интеллектуальному, контекстно-зависимому анализу. Логика, опробованная на фаззинге директорий, легко переносится и на другие области:

- **Фаззинг параметров:** Оптимизация работы таких инструментов, как x8, Arjun или ParamMiner, путем предсказания наиболее вероятных имен параметров на основе техдетекта.
- **Фаззинг поддоменов:** Генерация гипотетических поддоменов, релевантных для конкретной организации, на основе данных, найденных в открытых источниках.

Таким образом, будущее инструментов автоматизированной безопасности лежит в симбиозе классических методов и искусственного интеллекта, что позволяет специалистам сосредоточиться на сложных задачах анализа, а не на рутинном переборе значений.