DOI 10.37882/2223-2966.2026.02.49

ПОСТРОЕНИЕ РАСПРЕДЕЛЕННОЙ СИСТЕМЫ СЕТЕВОГО СКАНИРОВАНИЯ ВЕБ-ПРИЛОЖЕНИЙ С ФУНКЦИЯМИ СПАЙДЕРИНГА И КРАУЛИНГА

BUILDING A DISTRIBUTED WEB APPLICATION NETWORK SCANNING SYSTEM WITH SPIDERING AND CROWLING FUNCTIONS

G. Shipulin

Summary. The article is devoted to the consideration of issues related to the collection of information about the functioning of web applications in the framework of security assessment and penetration testing. The paper considered the main elements of the information collection process during penetration testing according to the OWASP Web Security Testing Guide web application penetration testing methodology, including the categories of data collected and aspects of their analysis for metadata, information about the components of the web application and the technology stack used. Based on the analysis of existing automated web scanning tools for web applications, their main limitations were identified (insufficient scanning coverage and completeness of data collection, mono tasking). The architecture of the developed distributed web application network scanning system with component-level spidering and crowling functions is described, and the results of its experimental launches are presented.

Keywords: penetration testing, OWASP, web application, network scanning, crawling, spidering, information collection, information security.

Шипулин Георгий Фаризович
Кандидат юридических наук, доцент, Российский технологический университет МИРЭА; Доцент, Московский Политехнический Университет
podumai_nad@mail.ru

Аннотация. Статья посвящена рассмотрению вопросов, связанных со сбором информации о функционировании веб-приложений в рамках проведения оценки защищенности и тестирования на проникновение. В работе были рассмотрены основные элементы процесса сбора информации при тестировании на проникновения согласно методике тестирования на проникновение веб-приложений OWASP Web Security Testing Guide, включая категории собираемых данных и аспекты их анализа на предмет наличия метаданных, информации о компонентах веб-приложении и используемом стеке технологий. На основе анализа существующих автоматизированных средств сетевого сканирования веб-приложений были выявлены их основные ограничения (недостаточность охвата сканирования и полноты сбора данных, монозадачность). Описана архитектура разработанной распределенной системы сетевого сканирования веб-приложений с функциями спайдеринга и краулинга на уровне компонентов, а также представлены результаты ее экспериментальных запусков.

Ключевые слова: тестирование на проникновение, OWASP, веб-приложение, сетевое сканирование, краулинг, спайдеринг, сбор информации, информационная безопасность.

Ввиду повсеместной цифровизации общественных отношений и использования веб-приложений в качестве средства оказания услуг вопросы оценки состояния и обеспечения защищенности веб-приложений являются актуальными.

Одним из подходов к оценке безопасности веб-приложения является тестирование на проникновение, позволяющий обнаружить актуальные уязвимости до их эксплуатации злоумышленникам. Основной методикой тестирования на проникновение веб-приложений является OWASP Web Security Testing Guide, в котором отдельно выделяется этап сбора информации об исследуемой системе [1].

Сбор информации о веб-приложении включает: получение информации о структуре и конфигурации веб-приложения, версии и типе используемого веб-сервера; поиск файлов метаданных для выявления скрытых элементов; получение информации о компонентах исследуемого веб-приложения, их составе и версиях, а также контента веб-страниц и файлов JavaScript; составление карта веб-приложения.

В рамках данного этапа, с одной стороны, требуется получить максимальное количество информации в сжатые сроки, что требует применения множества разных программных инструментальных средств автоматизации процесса сбора и анализа данных [1, 2]. Опыт использования таких инструментальных средств позволил сделать выводы об ограниченности применения при решении задач (большинство утилит монозадачны) и возникающих неудобствах использования, к которым относятся: необходимость конфигурации; отсутствие

единого стандартизированного вывода данных; отсутствие распределения нагрузки на исследуемую систему; преимущественно не реализованы отказоустойчивость, а также модуль генерации UserAget-ов, используемые при активном сетевом взаимодействии с исследуемой системой.

Таким образом, ввиду использования разных инструментальных средств в рамках одного этапа по сбору информации об исследуемой системе, возникает потребность в применении программного инструментального средства с реализованным функционалом, обеспечивающим решение большинства задач данного этапа [3].

Целями построения системы сетевого сканирования с функциями спайдеринга и краулинга являются сокращение времени на сбор информации об исследуемой системе и повышение качества выполнения данных процедур, выражающееся в большем количестве собираемой информации, необходимой для дальнейшей оценки защищенности и выполнения тестирования на проникновение исследуемого веб-приложения.

В связи с этим были сформированы функциональные характеристики, которые должны быть реализованы в системе сетевого сканирования веб-приложений с функциями спайдеринга и краулинга:

- составление «карты» веб-приложения (определение его структуры);
- определение базовой операционной системы, под управлением которой работает сервер, типа веб-сервера, компонентов веб-приложения, запущенных на веб-сервере, наличия и типа WAF, типа CMS, разрешенных HTTP-методов;
- поиск и загрузка пользовательских файловых вложений, а также доступных файлов веб-приложения прошлых или тестовых версий;
- агрегация и нормализация данных;
- распределение нагрузки и отказоустойчивости при работе системы;
- подстановка UserAgent-ов в тела запросов при активном сетевом взаимодействии с исследуемой системой.

Для достижения поставленных целей и задач была выбрана архитектура распределенной системы, основанная на модели Master-Worker [4], которая позволяет обеспечить выполнение определенного функционала по сбору и анализу данных.

Таким образом, система состоит из двух основных видов компонент: Master-контроллера и Worker-узлов, которых может быть несколько. При этом для взаимодействия оператора с системой был реализован интерфейс в виде отдельного сервиса для управления Master-контроллером, а для обеспечения взаимодействия между Master-контроллером и Worker-узлами, включая асинхронное распределение задач с контролем их выполнения — служебный интерфейс, реализованный с помощью библиотеки multiprocessing языка программирования Python.

При программной реализации описанной архитектуры системы использовался интерпретируемый язык программирования Python ввиду большого количества библиотек и фреймворков, реализующих поиск и обработку веб-страниц.

Интерфейс управления Master-контроллером реализован с помощью фреймворка Flask [5] языка программирования Python через который осуществляется: запуск сканирования (POST /scan); запрос статуса выполнения задания с возможностью его принудительного завершения, остановки и возобновления (GET /scan/<id>/progress); просмотр итогового отчета о выполненном задании с опцией его выгрузки в виде файла в JSON-формате (GET /scan/<id>).

Запуск сканирования осуществляется с указанием целевого домена для сканирования (параметр domain), ограничений по максимальной глубине обхода (параметр max_depth), числа сканируемых страниц (параметр max_pages), общего времени работы над задачей (параметр timeout) и максимального времени выполнения одного запроса (параметр request_timeout), а также количества Worker-узлов, которые будут осуществлять выполнение задачи (параметр workers). Данные параметры передаются Master-контроллеру.

Примеры запуска сканирования и отслеживания его процесса выполнения через интерфейс управления представлены на рисунке 1 и рисунке 2.

Master-контроллер, отвечающий за управление всем процессом сканирования и анализа, включает:

- модуль управления очередью URL (URL Frontier), реализующий алгоритм «поиска в ширину»;
- модуль диспетчеризации задач, который распределяет URL-адреса по свободным Worker-узлам;
- модуль агрегации, который собирает данные о веб-страницах, поступающие от Worker-узлов;
- модуль анализа, отвечающий за определение базовой операционной системы под управлением которой работает сервер, типа веб-сервера, компонентов веб-приложения, запущенных на веб-сервере, наличия и типа WAF, типа CMS, разрешенных HTTP-методов.

На основе анализа порядка заголовков в теле HTTP-ответа веб-приложения и их содержимого, структуры каталогов веб-приложения, файла robots.txt и пр. возможно определение ряда технических характеристик

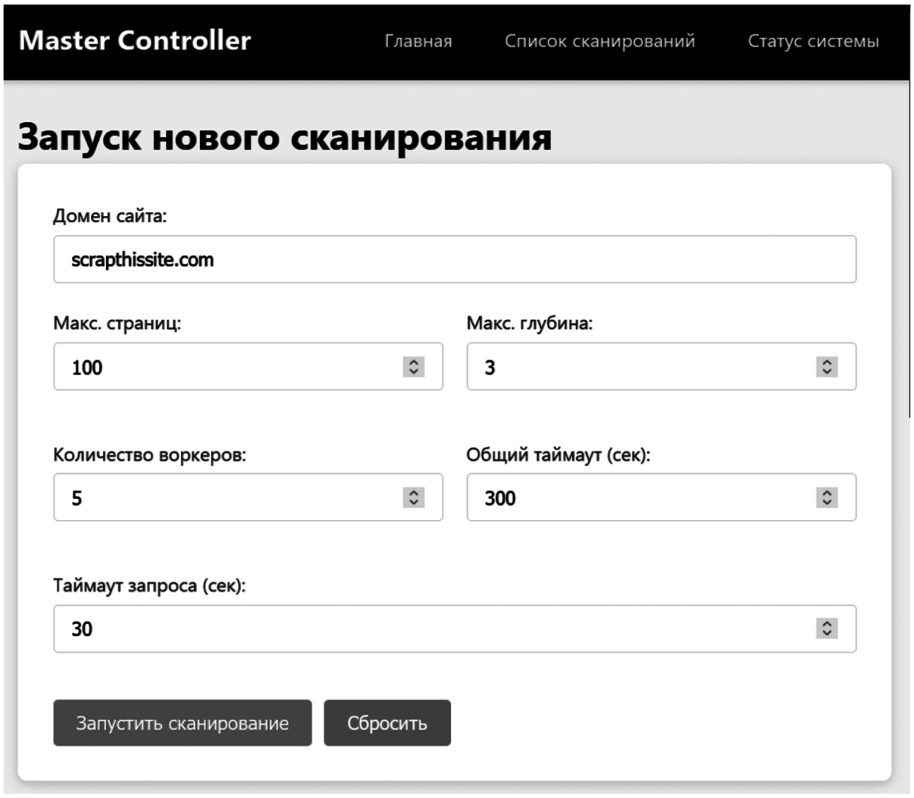

Рис. 1. Настройка параметров сканирования и его запуск (источник: разработка автора)

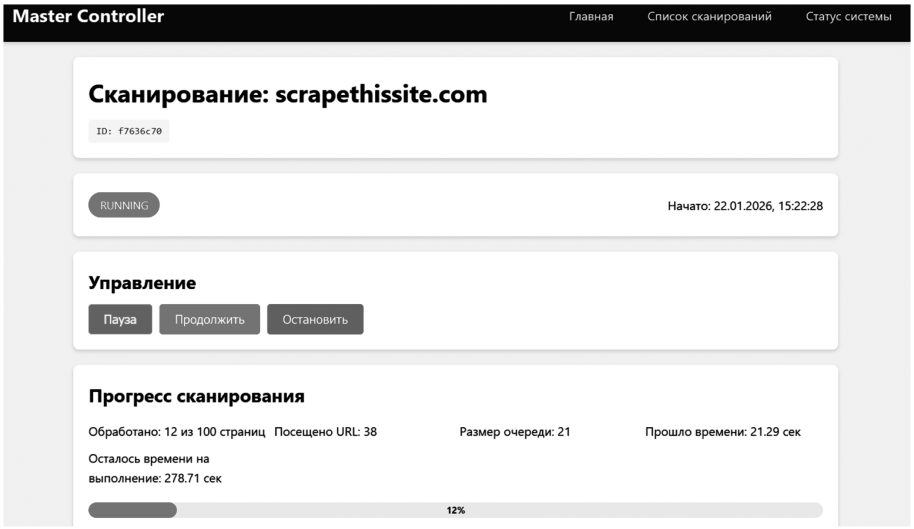

Рис. 2. Окно процесса сканирования (источник: разработка автора)

и параметров веб-приложения. Например, тип и версию CMS можно определить путём анализа структуры каталогов, которую каждая CMS выстраивает самостоятельно, или чтением файла robots.txt, в котором перечислен перечень разрешенных к индексированию веб-страниц поисковым системам. Также тип веб-сервера и базовую операционную систему, наличие WAF можно определить исходя из порядка заголовков в теле HTTP-ответа и его содержимого [1,6,7].

- модуль краулинга, отвечающий за загрузку доступных файлов, размещенных в веб-приложении. Загрузка файлов происходит на основе списка уникальных URL-путей к файлам, сформированного путем обработки (объединения, фильтрации, устранения дубликатов) списков путей к файлам, полученных от Worker-узлов.

Для контроля выполнения задач и отслеживания состояния очереди используются механизмы таймаутов (модуля multiprocessing), а через REST API-шлюз выполняется получение команд от оператора и отправка результатов выполнения задачи в JSON-формате.

Каждый Worker-узел реализует конвейер обработки одного URL-адреса, его работа состоит в: получении за-

дачи от Master-контроллера; создании объекта requests. Session() модуля requests языка программирования Python для установки значений заголовков User-Agent, используемых для эмуляции разных устройств при помощи функции setup_session() и установки таймаута выполнения запроса (request_timeout); выполнении сетевого запроса к целевому URL-адресу используя реализованную функцию process_url() и ранее созданный объект requests.Session().

После выполнения запроса к веб-странице по заданному URL-адресу с помощью метода get объекта requests. Session (), ответ сервера инкапсулируется в объект response, осуществляется определение MIME-типа по заголовку Content-Type при помощи реализованных функций detect_content_type_by_header () и get_resource_type (), а также проверка кода состояния HTTP-ответа при помощи метода raise_for_status () объекта response.

Для HTML-страниц посредством функций библиотеки BeautifulSoup языка программирования Python извлекаются все гиперссылки (теги <a href=...>) с последующей их нормализацией и группированием на внутренние и внешние [8,9].

В случае обнаружения текстовых файлов, файлов исходных кодов приложения (файлы административной панели, отладочных файлы, скрытые копии или прошлые версии), а также архивов, журналов событий, то информация об их месторасположении (полном пути в каталоге веб-приложения) передается модулю краулинга Master-контроллера.

В результате обработки ответа веб-приложения создается структурированный словарь (набор данных вида: «ключ»: «значение») с метаданными, который отправляется Master-контроллеру.

Каждый Worker-узел взаимодействует независимо с Master-контроллером, получая задачи и возвращая результаты, что обеспечивает разделение задач и отказоустойчивость системы.

Для проверки работоспособности реализованной системы и целесообразности выбора распределенной архитектуры была проведена серия сканирований разрешенного к тестированию публичного веб-приложения с доменным именем «scrapethissite.com», предназначенного прежде всего для тестирования систем спайдеринга и скрапинга.

В первом сканировании использовался общедоступный инструмент webscraper.io. в SaaS, во втором и третьем — использовалась разработанная система с одним и пятью активными Workers-узлами соответственно с заданием следующих количественных параметров сканирования: глубина сканирования (max_depth) — 3, максимальное количество сканируемых страниц (max_pages) — 38, максимальное время выполнения задачи (timeout) –100 секунд, максимальное время выполнения одного запроса (request_timeout) — 15 секунд. Результаты контрольных сканирований представлены в таблице 1.

Таблица 1.

Сравнительный анализ результатов проведенных сканирований

Метрика	Первое сканирование	Второе сканирование	Третье сканирование
Количество обработанных страниц	17	38	38
Количество посещенных уникальных URL-адресов	17	38	38
Общее время сканирования, сек.	58.12	61.47	17.12
Скорость обработки, стр./сек.	3.41	0.62	2.22
Количество найденных URL-ссылок	35	1454	1454
Количество страниц с ошибками (HTTP 4xx)	3	2	2

Источник: разработано автором.

Проведенный эксперимент наглядно демонстрируют эффективность разработанной системы. Созданная система при работе в однопоточном режиме (с одним активным Worker-узлом) обеспечила полный обход целевого ресурса (38 страниц, 1454 ссылки), но с меньшей скоростью обработки веб-страниц (0.62 стр./сек.) по сравнению с инструментом, примененном при первом сканировании. Ключевым результатом является существенное увеличение количества обрабатываемых запросов при масштабировании, поскольку использование пяти активных Worker-узлов позволило увеличить скорость обработки в 3.6 раза (до 2.22 стр./сек.), сократив общее время сканирования с 61.47 до 17.12 секунд.

Дополнительно, помимо составления карты веб-приложения было определено, что сканируемое веб-приложение использует базу данных PostgreSQL, запущено на веб-сервере Apache под управлением операционной системы Ubuntu Linux; не использует WAF; в качестве CMS используется WordPress. Таким образом, дополнительный функционал разработанной системы, позволяющий собрать большую часть требуемой информации об исследуемом веб-приложении, определяет ее преимущества применения.

Подводя итог, были рассмотрены основные элементы сбора информации о работе и конфигурации

веб-приложения в рамках проведения тестирования на проникновение, описаны архитектура и функционал разработанной системы сетевого сканирования с функциями спайдеринга и краулинга, включая определение: структуры веб-приложения, типа веб-сервера, компонентов веб-приложения, наличия и типа WAF, типа CMS, разрешенных HTTP-методов, поиска и выгрузки пользовательских файлов, агрегации и нормализации полученных данных. Дополнительными реализованными функциональными возможностями системы являются подстановка разных UserAgent-ов в телах запросов при активном сетевом взаимодействии с исследуемым веб-приложением, а также распределение задач между Worker-узлами. Использование данной системы целесообразно в первую очередь для организаций, выполняющих регулярную внутреннюю оценку защищенности и тестирование на проникновение веб-приложений.

ЛИТЕРАТУРА

1. OWASP Web Security Testing Guide // GitHub URL: https://github.com/OWASP/wstg?tab=readme-ov-file (дата обращения: 12.01.2026).
2. Web tools, или с чего начать пентестеру? // Habr URL: https://habr.com/ru/companies/dsec/articles/452836/ дата обращения: 12.01.2026).
3. Шипулин Г.Ф., Приймак А.Е. Использование техник спайдеринга и скрапинга при оценке состояния защищенности веб-приложений // Современная наука: актуальные проблемы теории и практики. Серия: Естественные и Технические Науки. — 2024. — №11. — С. 185–187. DOI 10.37882/2223–2966.2024.11.45.
4. Bogutskii A. Architecture of high-load distributed systems: the case of a web crawler / A. Bogutskii // Cold Science. — 2025. — No. 19. — P. 41–52.
5. Developing RESTful APIs with Python and Flask // Auth0 URL: https://auth0.com/blog/developing-restful-apis-with-python-and-flask/ (дата обращения: 12.01.2026).
6. Как рассказать о сайте поисковой системе // Habr URL: https://habr.com/ru/articles/761068/?ysclid=mkohqzqume110667024 (дата обращения: 12.01.2026).
7. Прессуем WordPress // Habr URL: https://habr.com/ru/articles/728294/?ysclid=mkohsc3e9l453188086 (дата обращения: 12.01.2026).
8. Beautiful Soup Tutorial — How to Parse Web Data with Python // Oxylabs URL: https://oxylabs.io/blog/beautiful-soup-parsing-tutorial (дата обращения: 12.01.2026).
9. Печников А.А. Адаптивный краулер для поиска и сбора внешних гиперссылок / А.А. Печников, Д.И. Чернобровкин // Управление большими системами: сборник трудов. — 2012. — № 36. — С. 301–315.

Журнал «Современная наука: актуальные проблемы теории и практики»