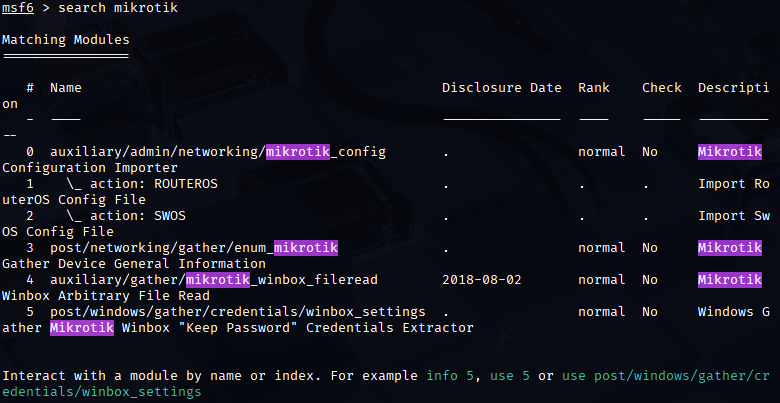
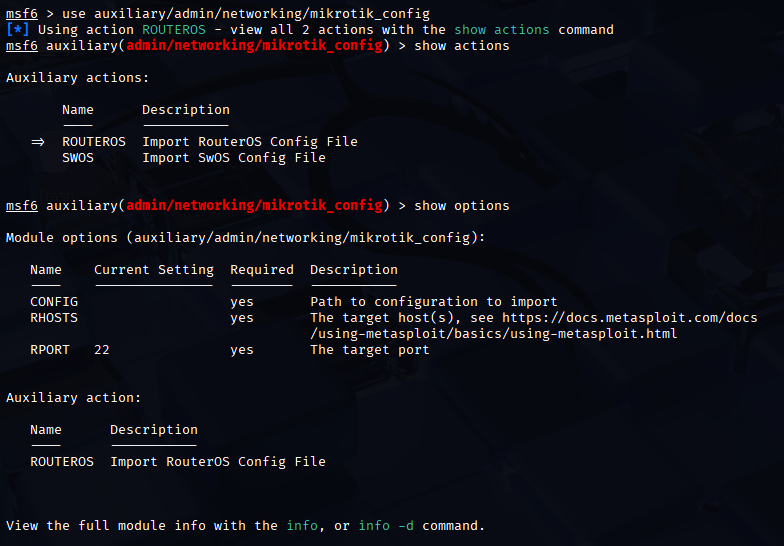
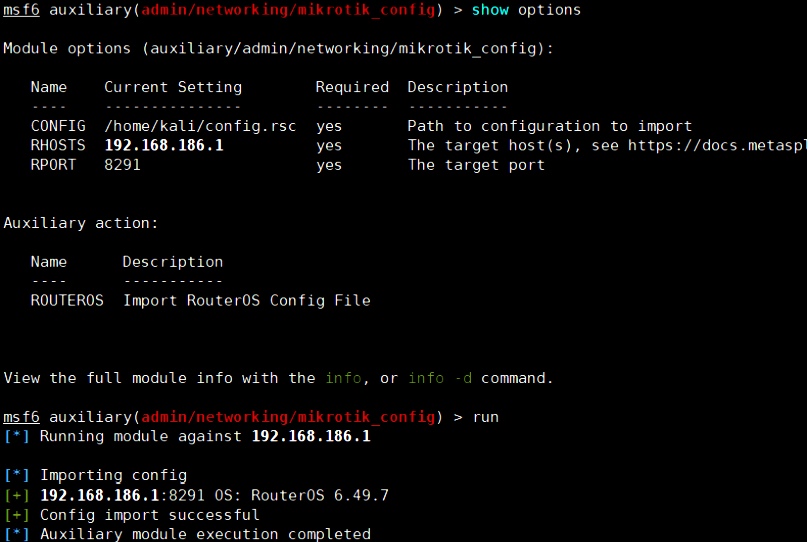
Развернем Kali Linux и запустим утилиту MSFconsole:









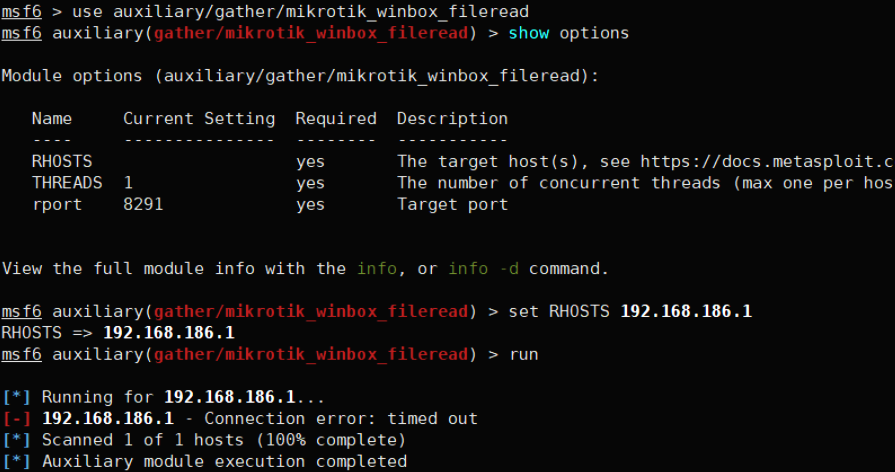
Что происходит? Эксплойт:

-Читает файл конфигурации MikroTik: Модуль анализирует файл конфигурации (например, config.backup), который указали.

-Импортирует данные в базу данных Metasploit: Модуль извлекает информацию из файла конфигурации и сохраняет её в базе данных Metasploit для дальнейшего анализа.

-Выводит информацию: Модуль отображает найденные данные (например, настройки WiFi) в терминале.

Запустим другой эксплойт, связанный с MikroTik:



Этот эксплойт Metasploit используется для чтения файлов на устройствах MikroTik, используя уязвимость CVE-2018-14847 (https://bdu.fstec.ru/vul/2018-01357). Эта уязвимость позволяет злоумышленнику обойти аутентификацию и читать файлы на устройстве через протокол WinBox.

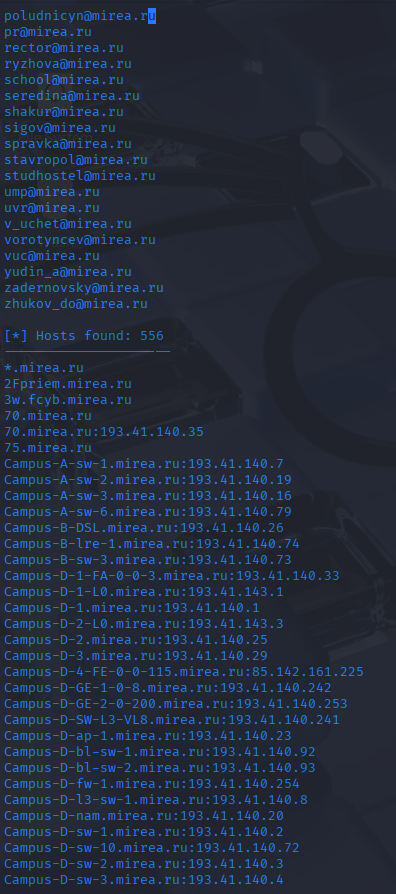
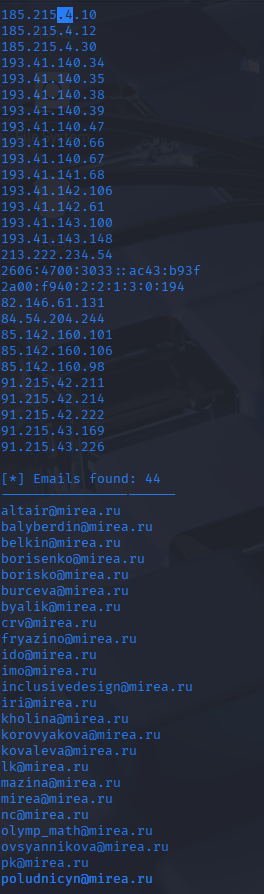
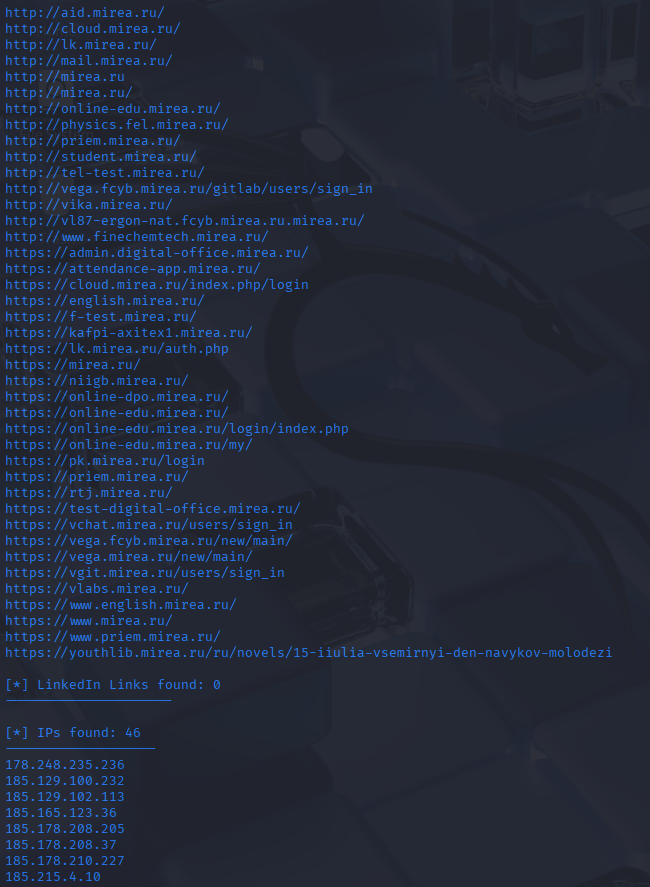
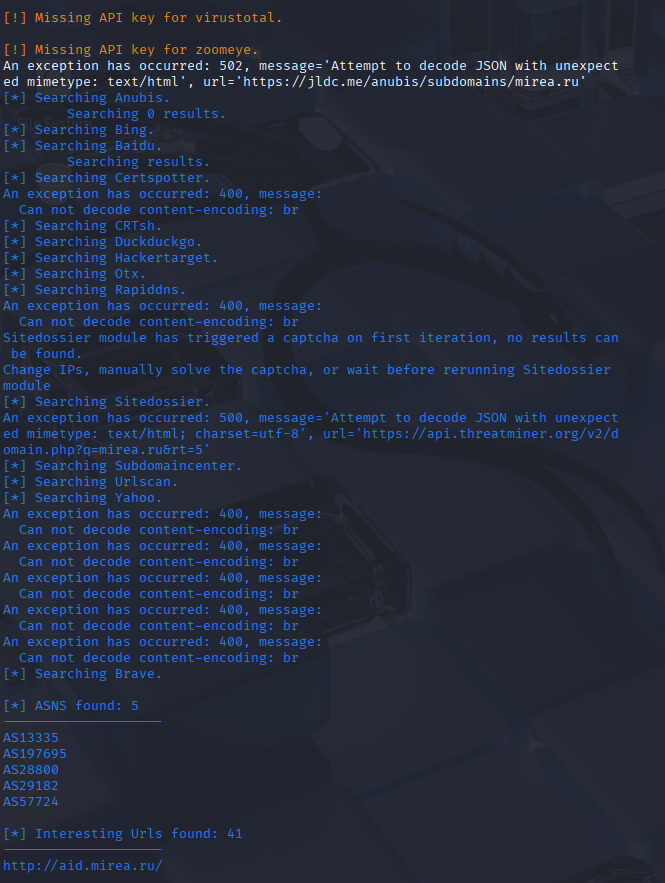
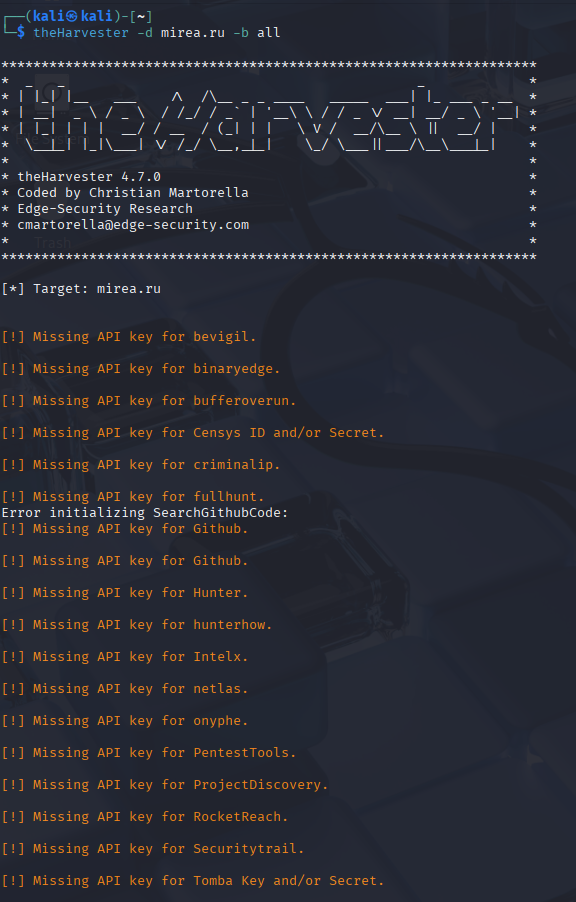
Mikrotik RouterOS до версий 6.42.7 и 6.40.9 уязвима для переполнения буфера стека через интерфейс обновления лицензии. Эта уязвимость теоретически может позволить удалённому злоумышленнику с подтверждённой учётной записью выполнить произвольный код в системе.

Как работает: обходом ограничений на доступ к файловой системе через интерфейс Winbox. Эта уязвимость позволяет удалённым неаутентифицированным атакующим читать произвольные файлы, а аутентифицированным — записывать их, что может привести к получению полного контроля над устройство.

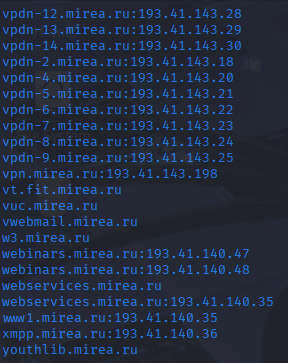
Уязвимость основана на недостаточной проверке путей к файлам (directory traversal) в протоколе Winbox (8291). Атакующий может отправить специально сформированный запрос, содержащий последовательности ../, чтобы выйти за пределы ограниченного каталога и получить доступ к чувствительным файлам, таким как база данных пользователей (user.dat) .

OSINT:

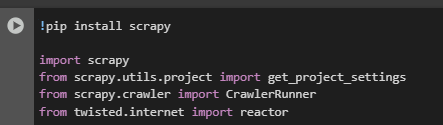
Hosts - поддомены, опечатка похоже



//…///

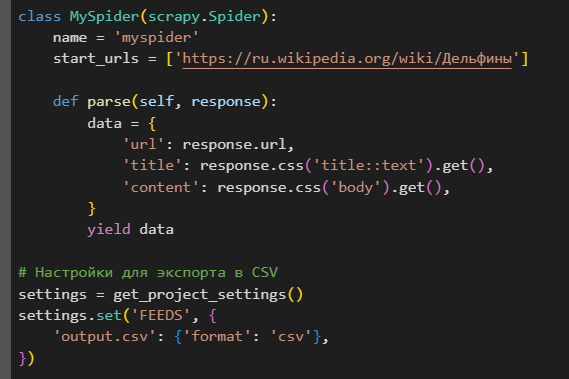


## Задание 1. Разработать модуль spider для получения данных с веб-приложений в формализованном виде



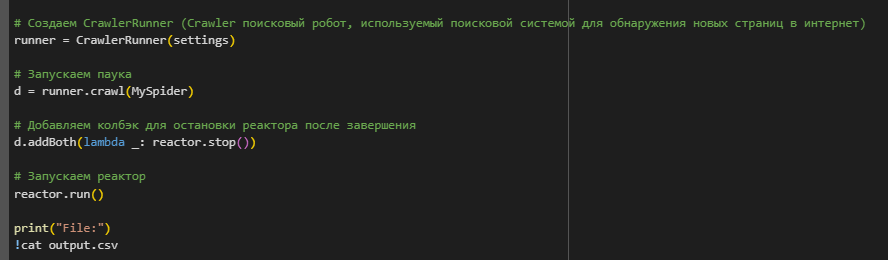
Устанавливаем scrapy и подключаем библиотеки, которые нам понадобятся

scrapy — основной инструмент для создания "пауков" (специализированных программ для скрейпинга (получения веб-данных путём извлечения их со страниц веб-ресурсов) сайтов)  
get\_project\_settings — позволяет получить настройки проекта Scrapy  
CrawlerRunner — запускает паука внутри Python-скрипта без отдельной консольной команды  
reactor — цикл событий из библиотеки Twisted, который управляет асинхронной работой (ожиданием ответов от сайтов)

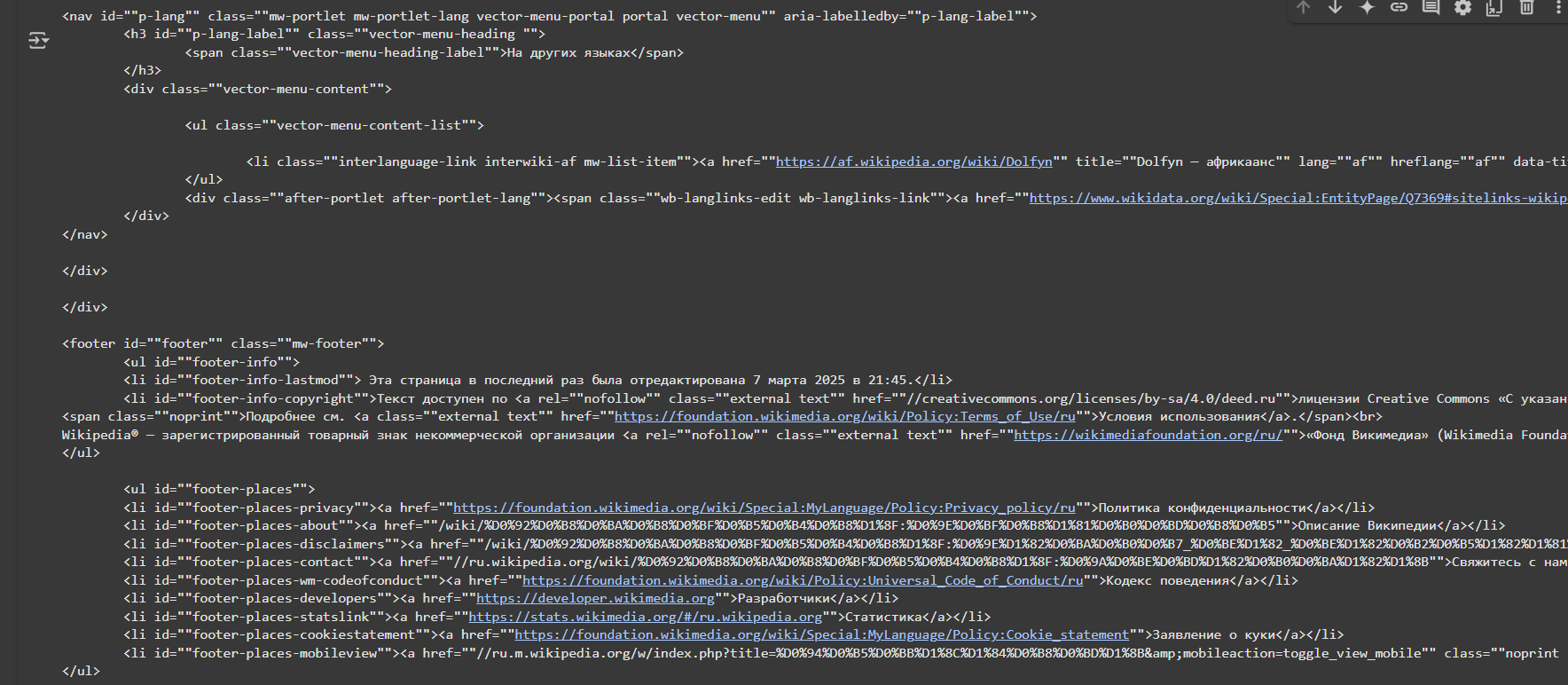


Далее делаем класс моего пивука, в котором указываем куда он полезет и что (URL, заголовок страницы и всё содержимое тела сайта) соберет

Потом указываем как будем сохранять



Тут создается объект, который будет управлять запуском паука с заданными настройками, запускается паук, указываем, что при после завершения работы паука останавливается цикл события reactor, запускается асинхронный цикл событий, чтобы всё работало. Выводим что там в файле аутпут.

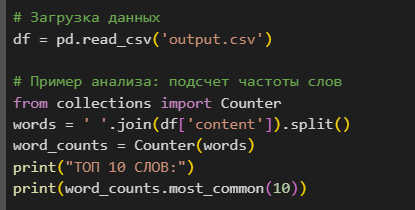


## Задание 2. Подготовить датасет и произвести его анализ

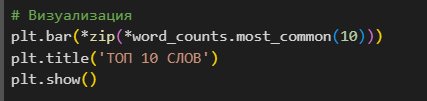


pandas (pd) — для удобной загрузки и обработки табличных данных.

matplotlib.pyplot (plt) — для построения графиков и визуализации данных.



Загружаем датасет, из content берем текст всех страниц, объединяя в одну строку, разбиваем на слова, подсчитываем, сколько раз каждое слово встречается.



Выводится список из 10 слов, которые чаще всего встречались в содержимом сайтов.

