Установка и настройка Apache  
Скачиваем Apache



Проверяем работает ли веб-сервер



Проверяем локальную доступность



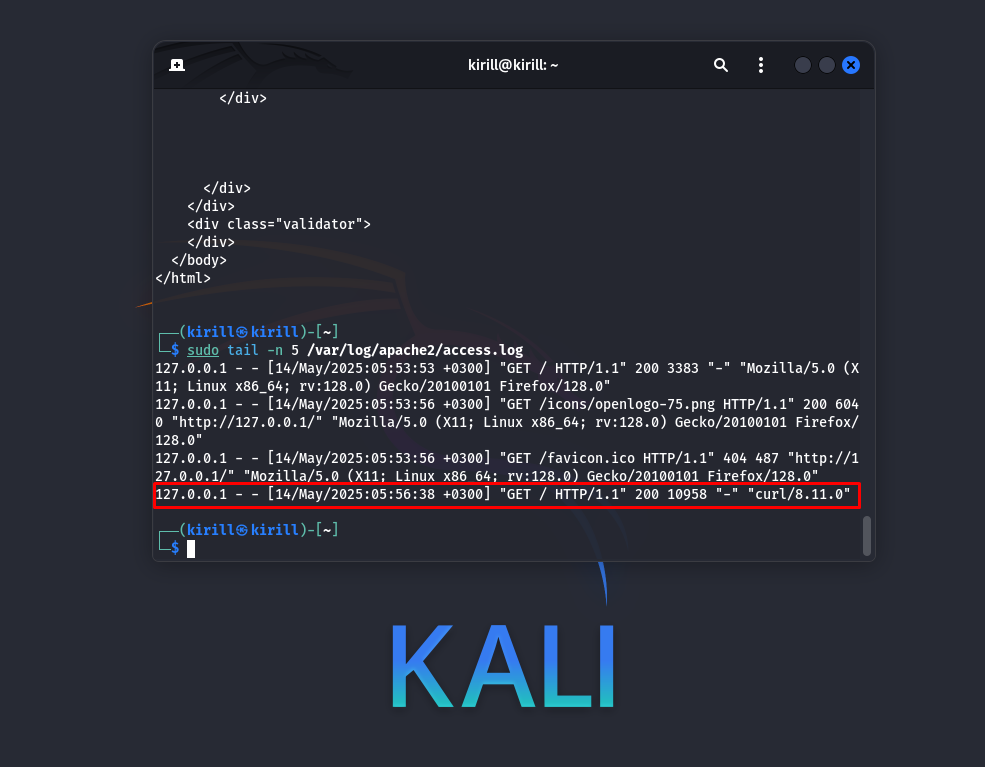
Как видно всё отлично

Настройка логирования

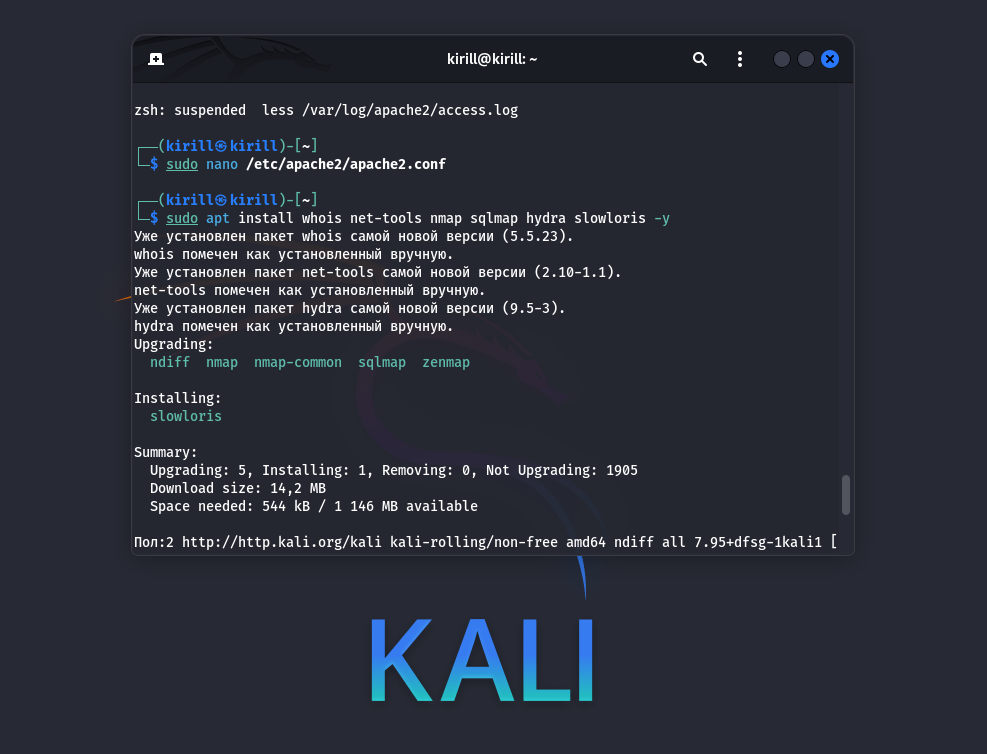
Запрашиваем страницу



Смотрим появился ли лог

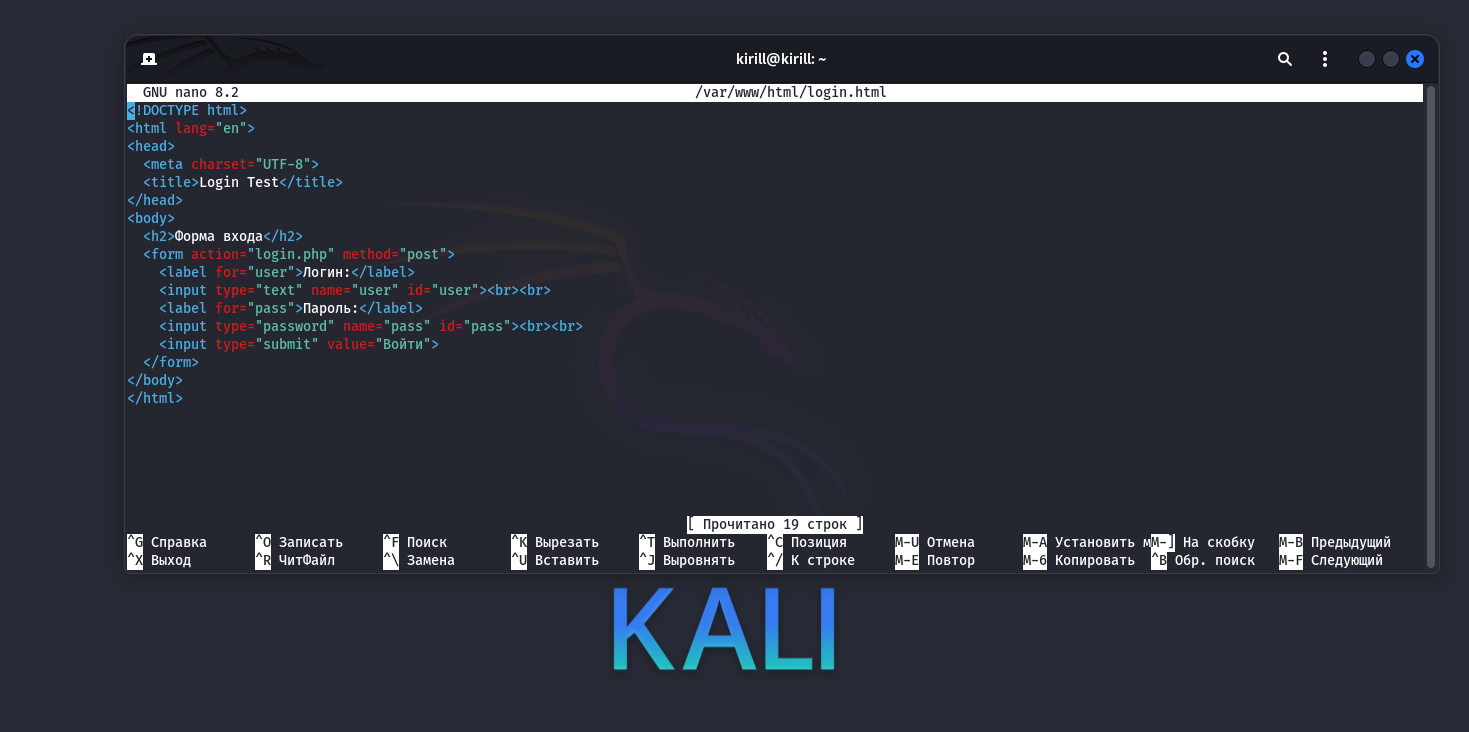


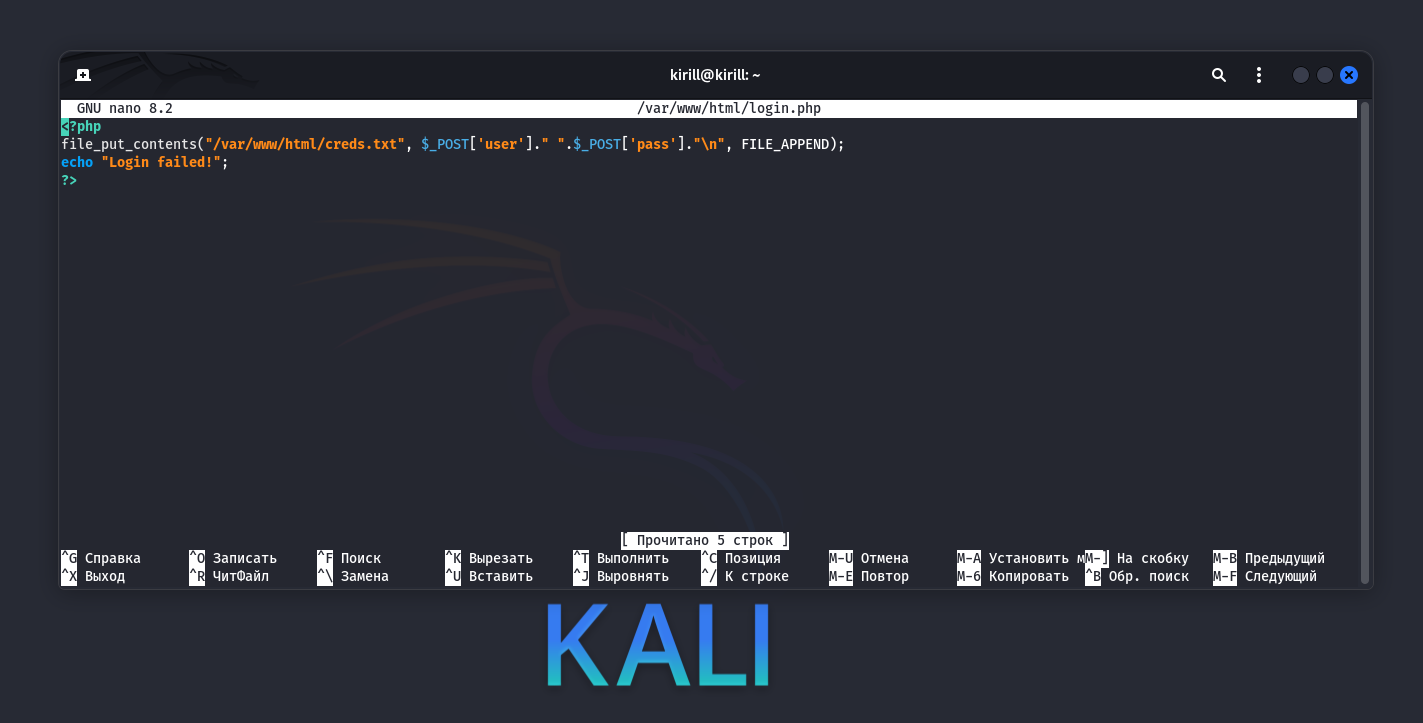
Установка инструментов для анализа



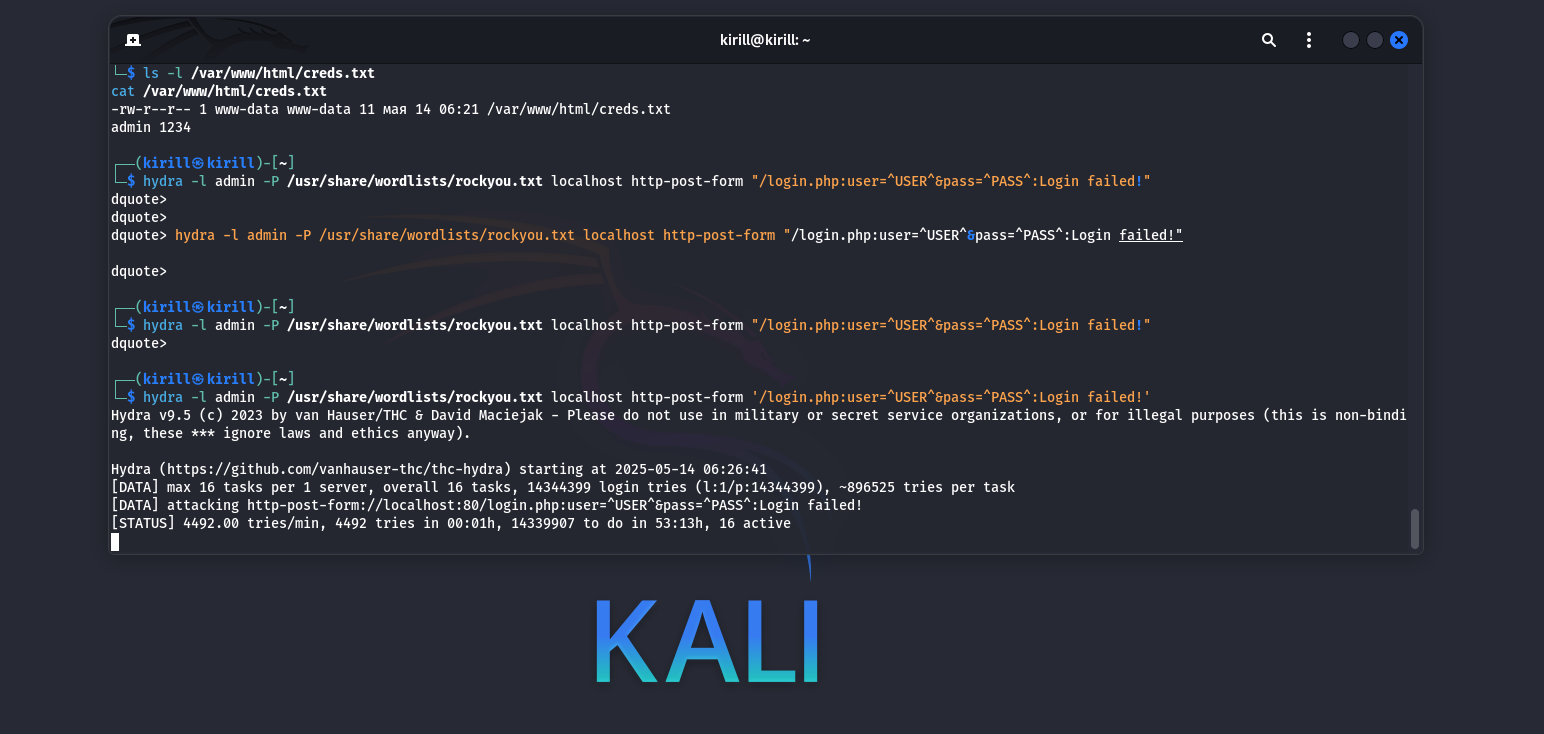
Brute-force атака

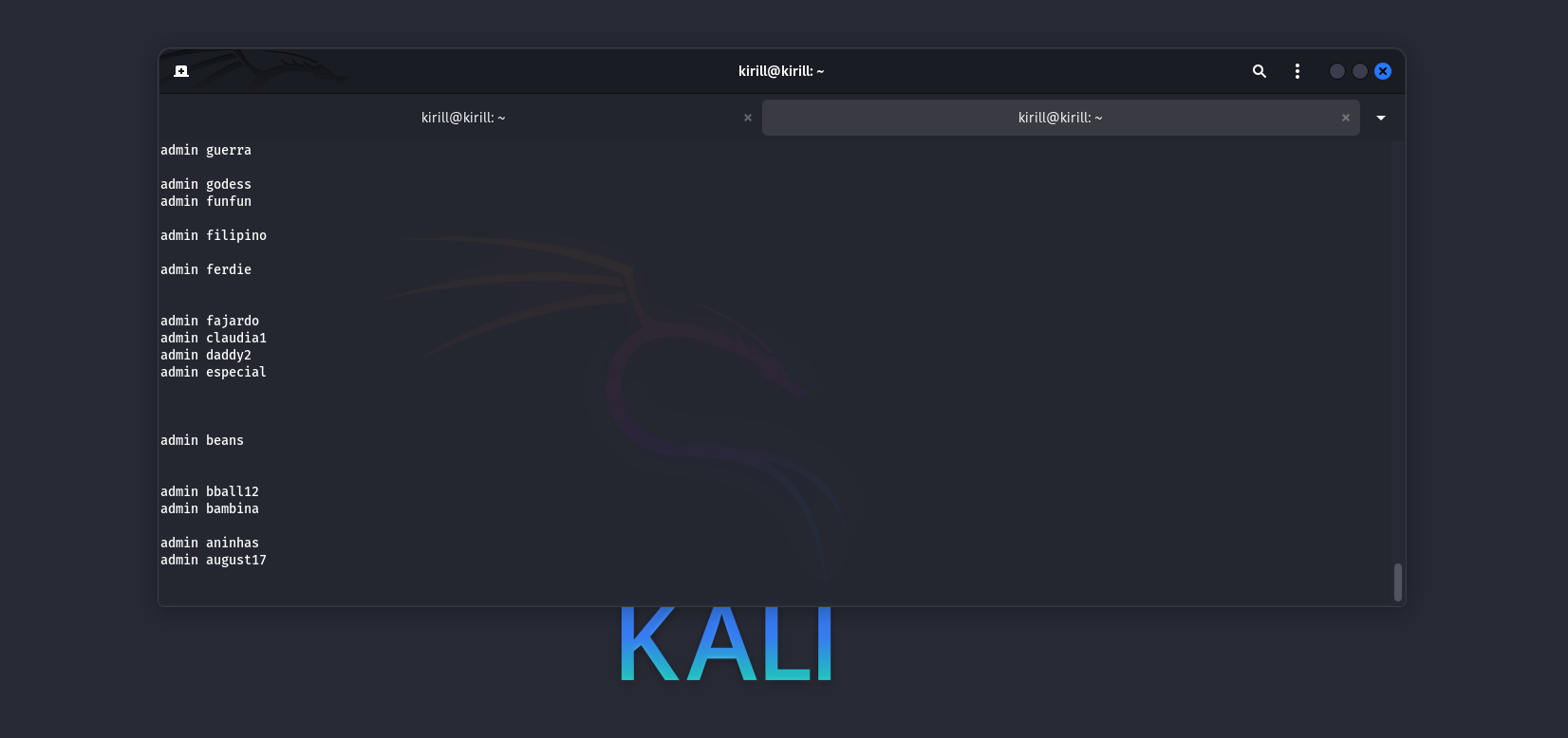
Создаём простую форму входа на сайтие





Запуск атаки

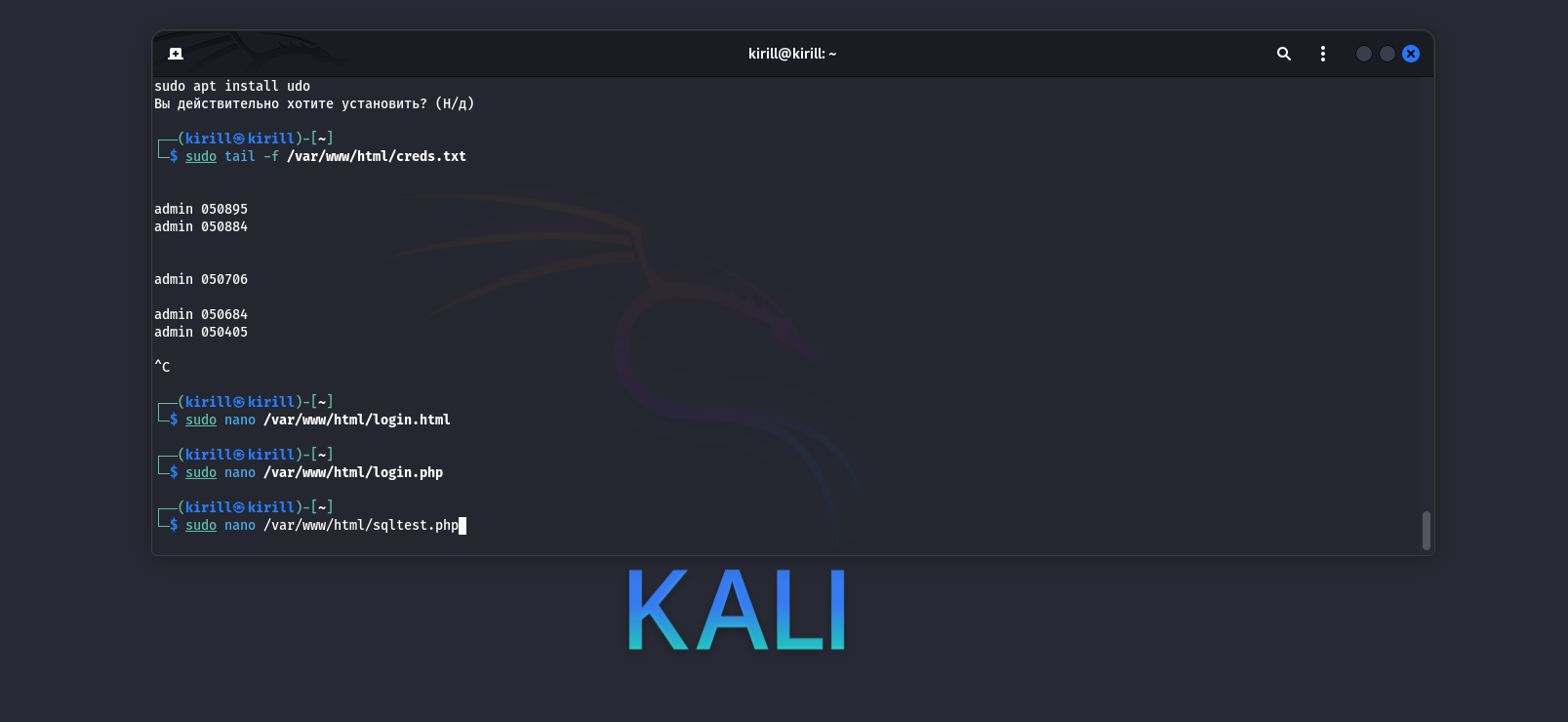


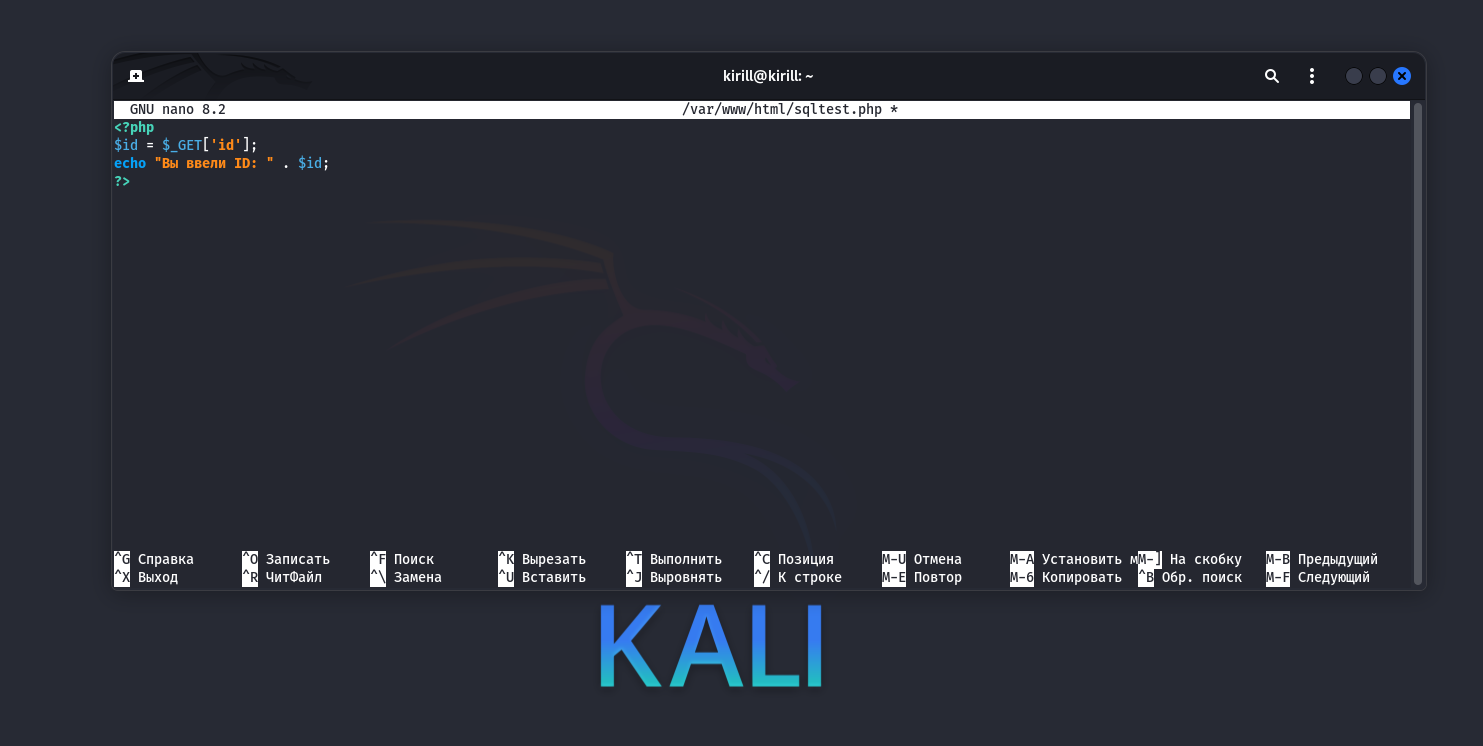


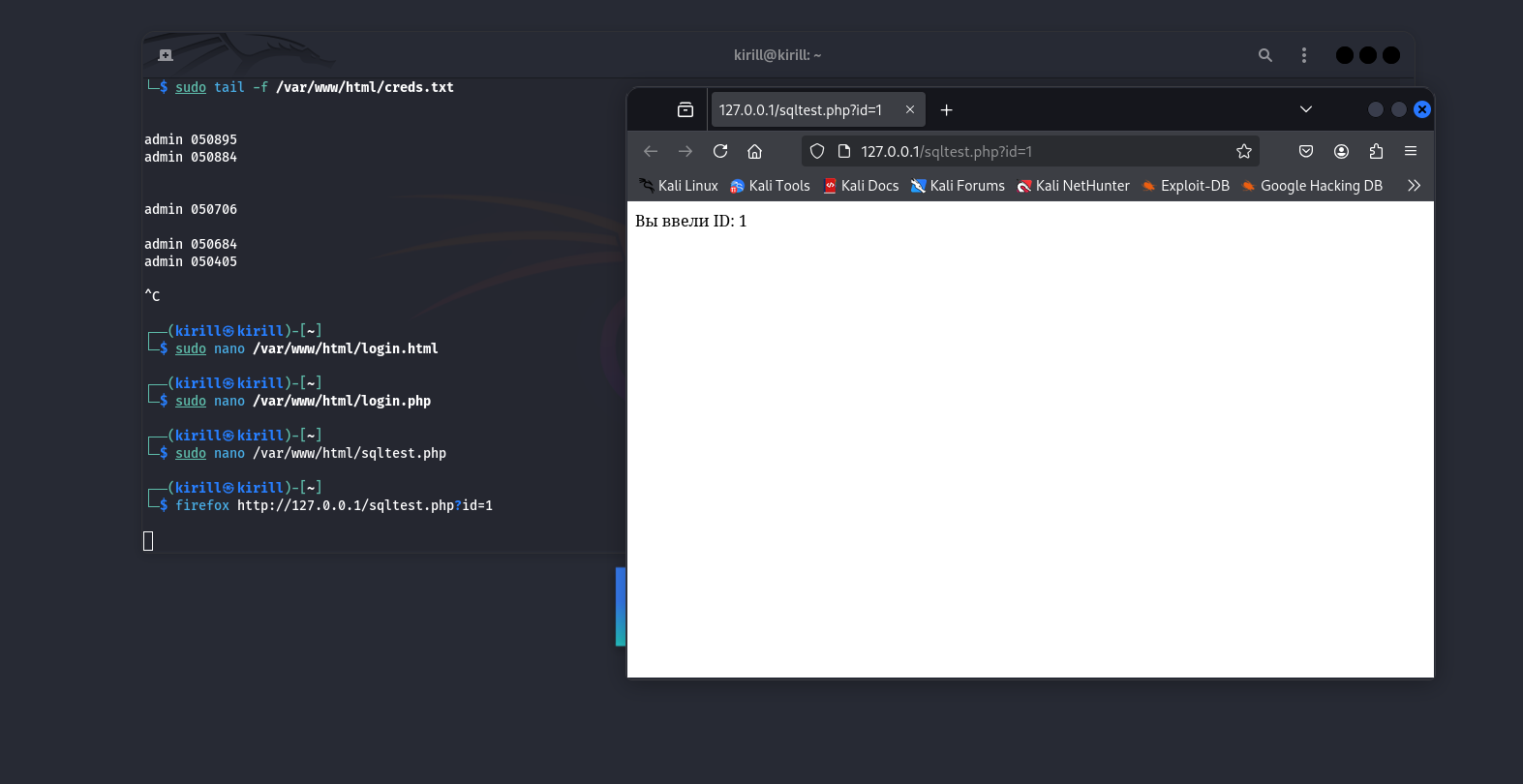
| **Поле** | **Значение** |
| --- | --- |
| **Тип инцидента** | Brute-force (перебор паролей через POST) |
| **Дата/время** | 14 мая 2025, 06:26 |
| **IP-адрес** | 127.0.0.1 |
| **Цель запроса** | /login.php |
| **Метод** | POST |
| **Инструмент** | hydra |
| **User** | admin |
| **Пароль найден?** | Да, через словарь rockyou.txt |
| **Лог Apache** | /var/log/apache2/access.log |
| **Признаки** | Много повторяющихся POST-запросов с разными паролями |
| **Контрмеры** | Ограничение количества попыток входа, установка fail2ban, Captcha |

SQL-инъекция

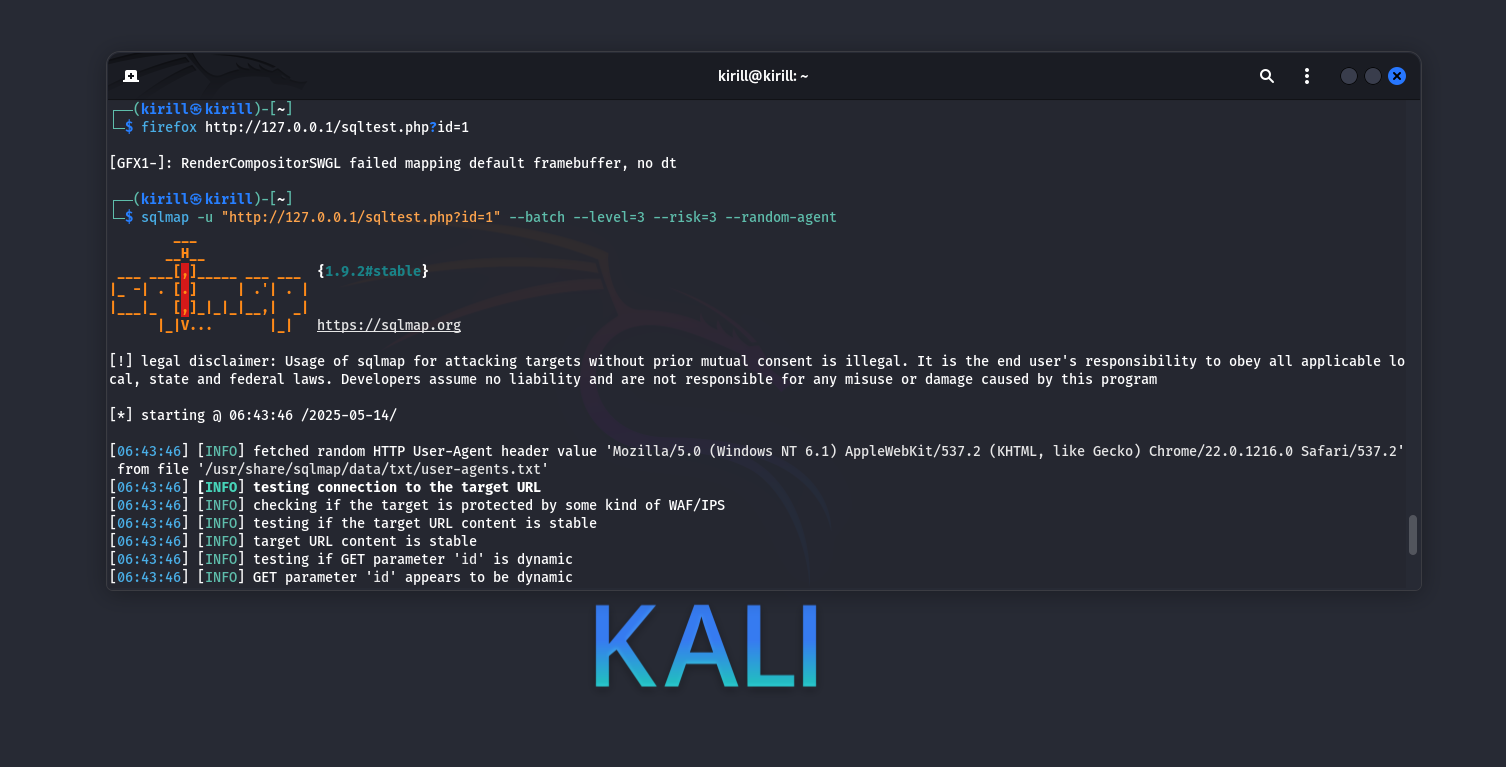
Создаём уязвимый скрипт







Эмулируем атаку





--batch — не задавать вопросы

--level=3 --risk=3 — расширенный анализ

--random-agent — имитирует разные браузеры

SQLmap начнёт проверку уязвимостей. Даже если реальной БД нет — он всё равно будет писать в лог **попытки SQL-инъекции** (через URL-параметры)

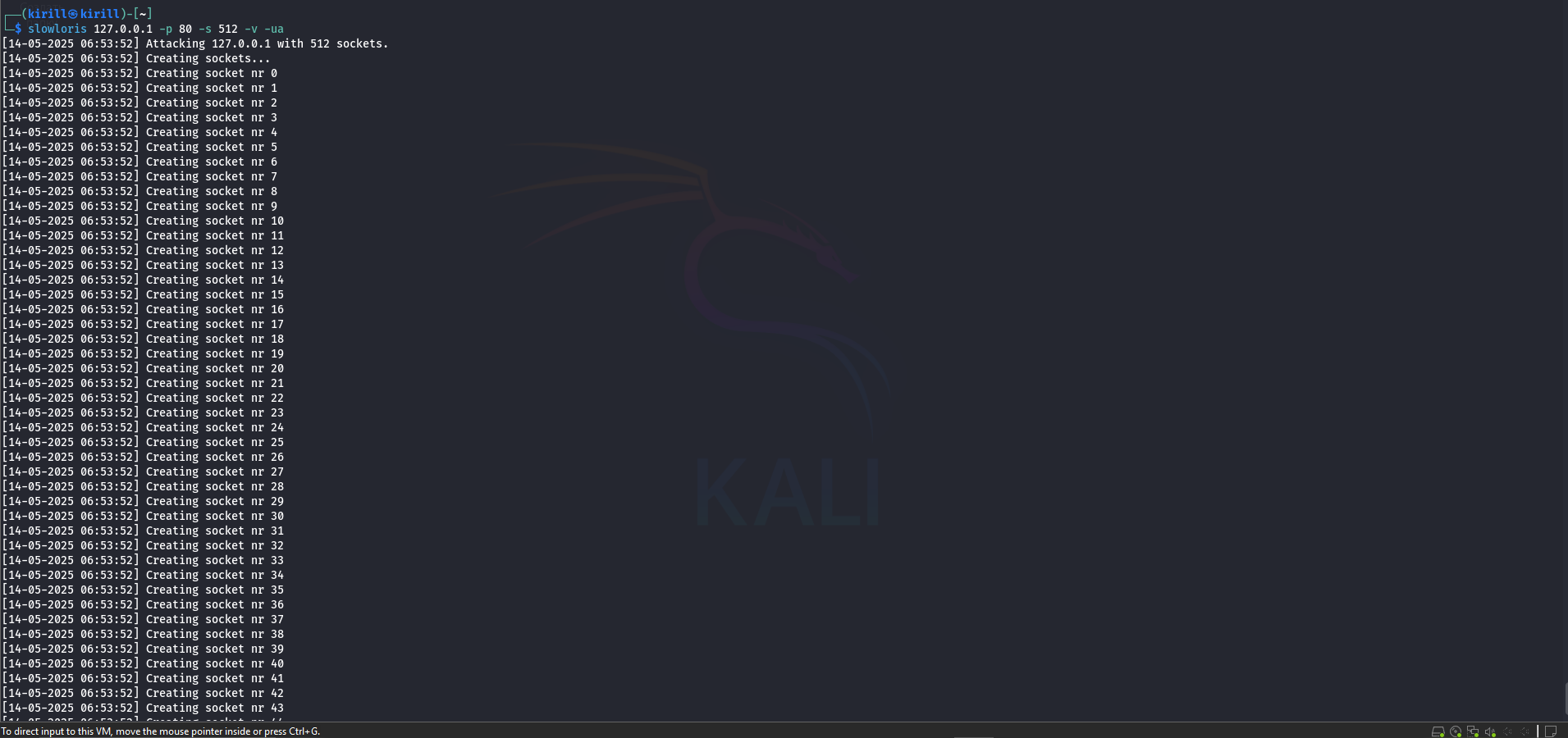


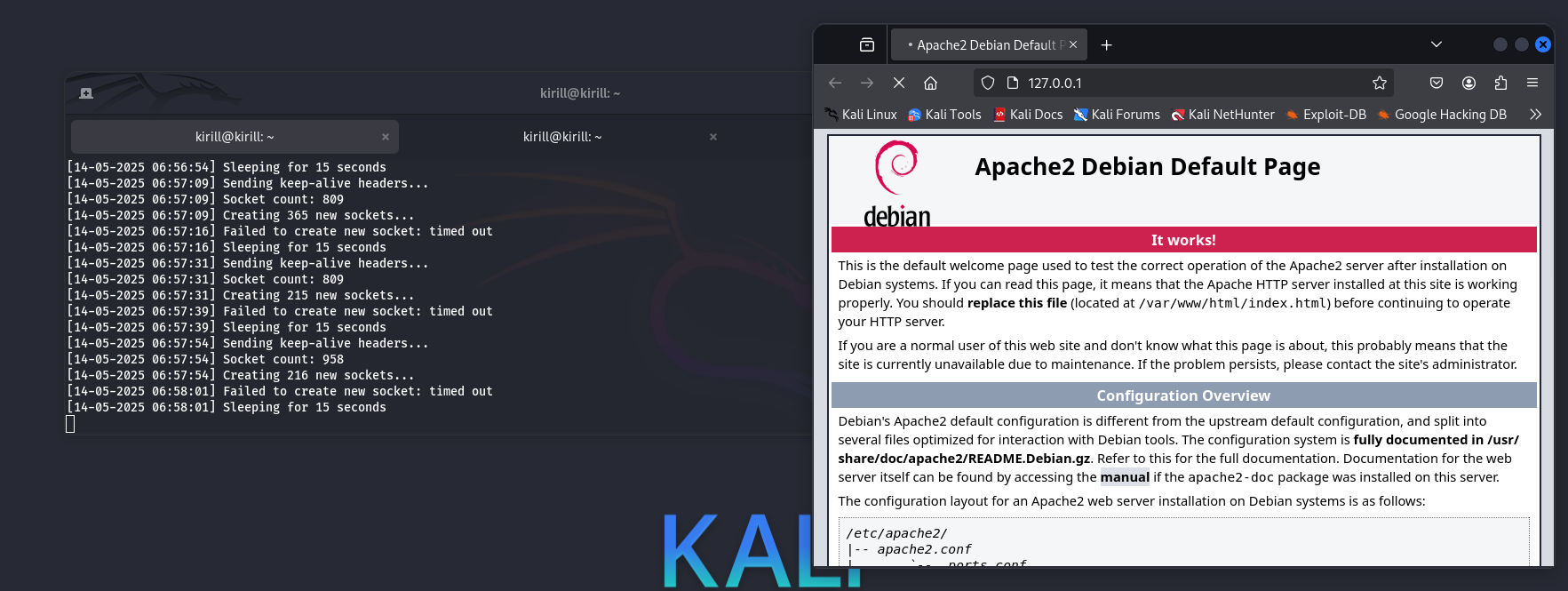


| **Поле** | **Значение** |
| --- | --- |
| **Дата/время** | 14 мая 2025, 06:44 |
| **IP-адрес** | 127.0.0.1 (локальный тест) |
| **Цель запроса** | /sqltest.php?id=1' ORDER BY X# |
| **Тип инцидента** | SQL-инъекция (SQL Injection — Injection testing via ORDER BY) |
| **Метод запроса** | GET |
| **User-Agent** | Mozilla/5.0 ... Chrome/22 ... |
| **Источник** | sqlmap |
| **Лог Apache** | /var/log/apache2/access.log |
| **Меры реагирования** | Фильтрация параметра, валидация чисел, использование prepared statements |

DoS-Атака

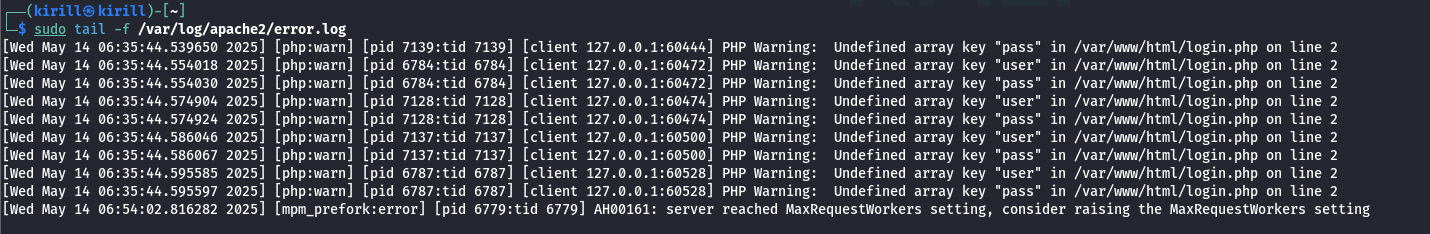
Запускаем атаку





Страница бесконечно пытается обновиться, значит атака работает

Логи:

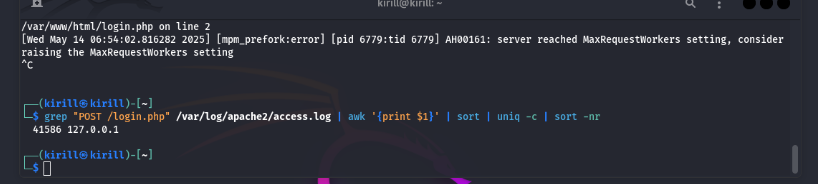


| **Поле** | **Значение** |
| --- | --- |
| Время | 14 мая 2025, 06:35 |
| Тип атаки | DoS (Slowloris) |
| Инструмент | slowloris |
| Признаки | MaxRequestWorkers reached, висящие PHP-запросы |
| Ошибки | Undefined array key "user" — запросы без тела |
| Метод реагирования | Установка mod\_reqtimeout, увеличение MaxRequestWorkers |

Фильтрация атак

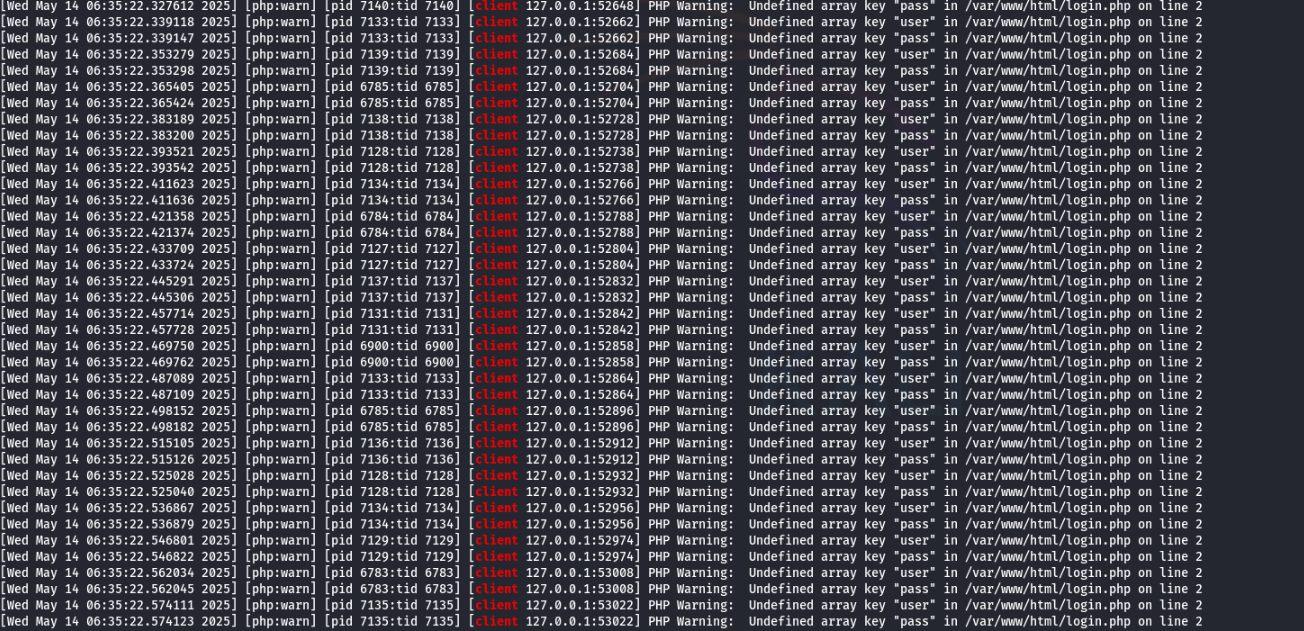
Brute-force атаки (на login.php):

grep "POST /login.php" /var/log/apache2/access.log | awk '{print $1}' | sort | uniq -c | sort –nr



SQL-инъекции (через sqltest.php)

grep "sqltest.php" /var/log/apache2/access.log | grep -Ei "union|select|or|and|order|--|%27|%22"

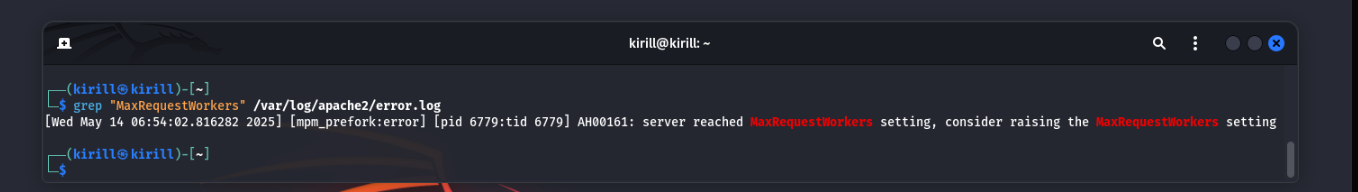


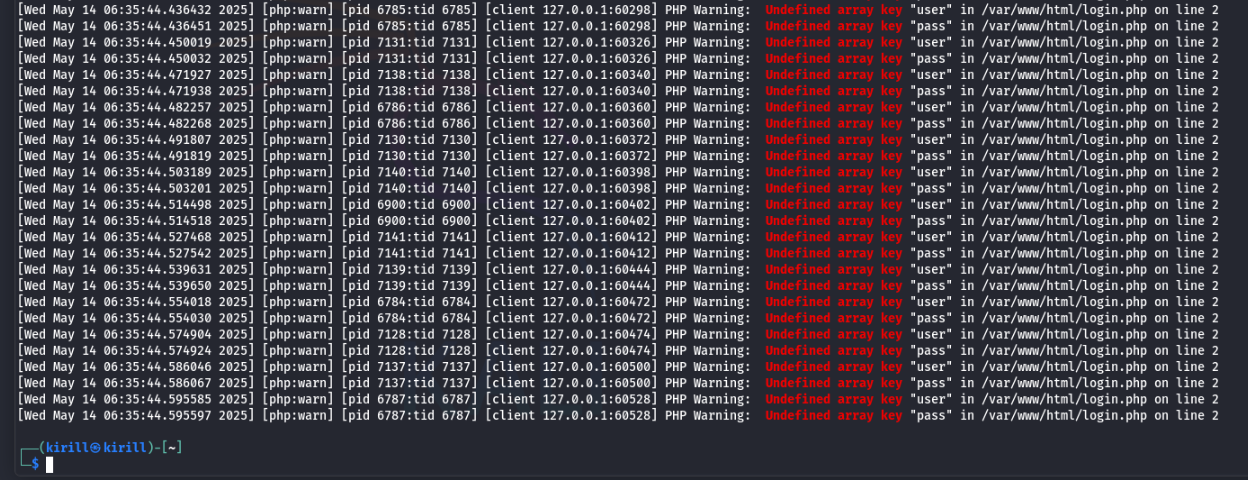
DoS-атаки (Slowloris)

grep "MaxRequestWorkers" /var/log/apache2/error.log

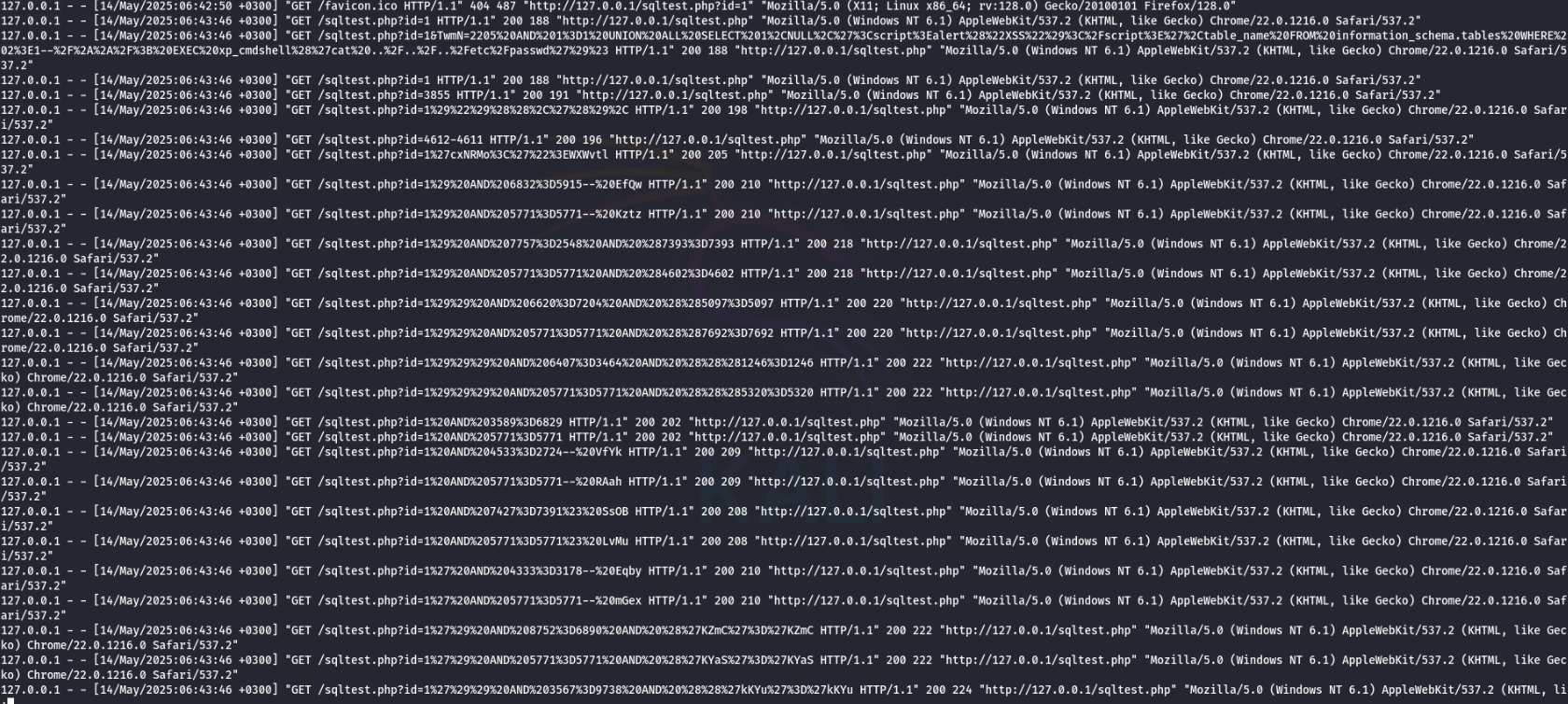
и

grep "Undefined array key" /var/log/apache2/error.log





Общая фильтрация всех GET с подозрительными параметрами



| **Тип атаки** | **Команда** |
| --- | --- |
| Brute-force | grep "POST /login.php" + `awk '{print $1}' |
| SQL-инъекция | grep "sqltest.php" + `grep -Ei "union |
| DoS (L7) | grep MaxRequestWorkers /var/log/apache2/error.log |
| PHP-ошибки | grep "Undefined array key" /var/log/apache2/error.log |

**Заключение**

В рамках выполненного задания была реализована система базовой безопасности веб-сервера Apache с упором на логирование и анализ ИБ-событий.

В ходе работы были достигнуты следующие результаты:

Установлен и настроен веб-сервер Apache в изолированной среде (локальный сегмент 127.0.0.1);

Включено ведение логов доступа и ошибок (access.log и error.log);

Проведены три успешные имитации атак:

**Brute-force** с использованием утилиты hydra;

**SQL-инъекция** через sqlmap с реальной базой данных SQLite;

**DoS-атака (Slowloris)** с перегрузкой Apache;

Получены и проанализированы реальные лог-файлы, отражающие суть каждой атаки;

Оформлены структурированные карточки ИБ-инцидентов, содержащие:

дату и время;

тип и цель атаки;

инструмент и IP-источник;

логи и рекомендации по защите.

Предложены меры реагирования и повышения устойчивости сервера:

Использование prepared statements в коде;

Ограничение количества соединений и попыток;

Внедрение модулей mod\_reqtimeout, fail2ban, WAF.

Таким образом, поставленная цель — **настройка системы логирования и базового анализа событий безопасности веб-сервера** — была успешно достигнута, задачи выполнены в полном объёме, результаты задокументированы и подтверждены логами и практическими тестами.