Розвідка та перерахування (опціонально)

Ці інструменти можна використовувати для ідентифікації поверхневої та контекстної інформації.

whois - Використовується для ідентифікації власника домену та інформації про реєстратора.

Використання:

whois example.com

whois example.com | less

dig – Використовується для перевірки DNS-записів та ідентифікації потенційних субдоменів або відкритих сервісів.

Usage:

dig example.com ANY

Деякі корисні запити

dig example.com A

dig example.com MX

dig example.com TXT

dig example.com NS

theHarvester - Використовує загальнодоступні джерела (Google, Bing тощо) для ідентифікації публічно проіндексованих електронних листів, хостів та доменів, пов’язаних із цільовим сайтом.

Використання:

theHarvester -d example.com

Перевірка багатьох джерел (це може зайняти деякий час):

theHarvester -d example.com -b all

amass – Використовується для активного та пасивного виявлення субдоменів.

Використання:

amass enum -passive -d example.com

Активний(агресивний) режим:

Amass enum -active -d example.com

Вивід результату у файл:

amass enum -passive -d example.com -o subs.txt

whatweb - Використовується для ідентифікації веб-технологій, коли HTTP-сервіси доступні.

Використання:

whatweb https://apiaddress:8443

Для цього сервера стандартні методи розвідки можуть дати обмежені або нульові результати через відсутність веб-інтерфейсу, наявність лише відкритих кінцевих точок API та наявність лише POST-запитів.

Сканування портів, скриптова перевірка та тестування вразливостей

HTTP/TLS конфігурації (nikto)

Nikto - використовується для виявлення поширених помилок конфігурації веб-сервера та небезпечних HTTP-заголовків.

Використання:

nikto -h https://apiaddress:8443 -ssl

Перевіряє:

відкриті admin-панелі

backup файли

небезпечні headers

Через API-орієнтовану архітектуру та відсутність HTML-сторінок результати Nikto обмежуються перевірками на рівні протоколу.

Сканування портів і сервісів (nmap)

Nmap – використовується для визначення відкритих портів, сервісів і TLS-конфігурації.

Базове використання:

nmap example.com

Перевірка конкретного порту:

nmap -p 8443 example.com

З визначенням версій використаних технологій (стара версія – потенційні вразливості):

nmap example.com -sV

Тестування з безпечними скриптами:

nmap example.com -sC

HTTPS-порт і HTTP-методи:

nmap -p 8443 --script http-headers, http-methods, http-enum example.com

Перевірка TLS-версії та на слабкі шифри:

nmap -p 8443 --script ssl-cert, ssl-enum-ciphers example.com

Перевірка порту бази даних:

nmap -p 3306 --script mysql-info, mysql-ssl example.com

Розширене сканування (виявлення ОС і сервісів, скрипти):

nmap -A example.com

Сканування зі скриптами пошуку вразливостей:

nmap –script vuln example.com

Для кожного сервісу:

- ➢ визначає тип сервісу
- ➢ запускає **vulnerability scripts**
- ➢ перевіряє відомі CVE

Перевіряє такі вразливості:

- ➢ старі версії Apache / Nginx
- ➢ слабкі SSL/TLS
- ➢ відомі SMB / FTP / SSH CVE
- ➢ небезпечні конфігурації

Перевірка сертифікату:

nmap -p 8443 --script ssl-cert example.com

Пошук CVE (nessus)

Nessus - пропрієтарний сканер вразливостей, використовується для співставлення знайдених сервісів і TLS-конфігурацій з відомими CVE.

1. Перейдіть на https://www.tenable.com/downloads/nessus?loginAttempted=true, виберіть Platform -> Linux, Debian, AMD64. Натисніть download.
2. Встановлення: в консолі

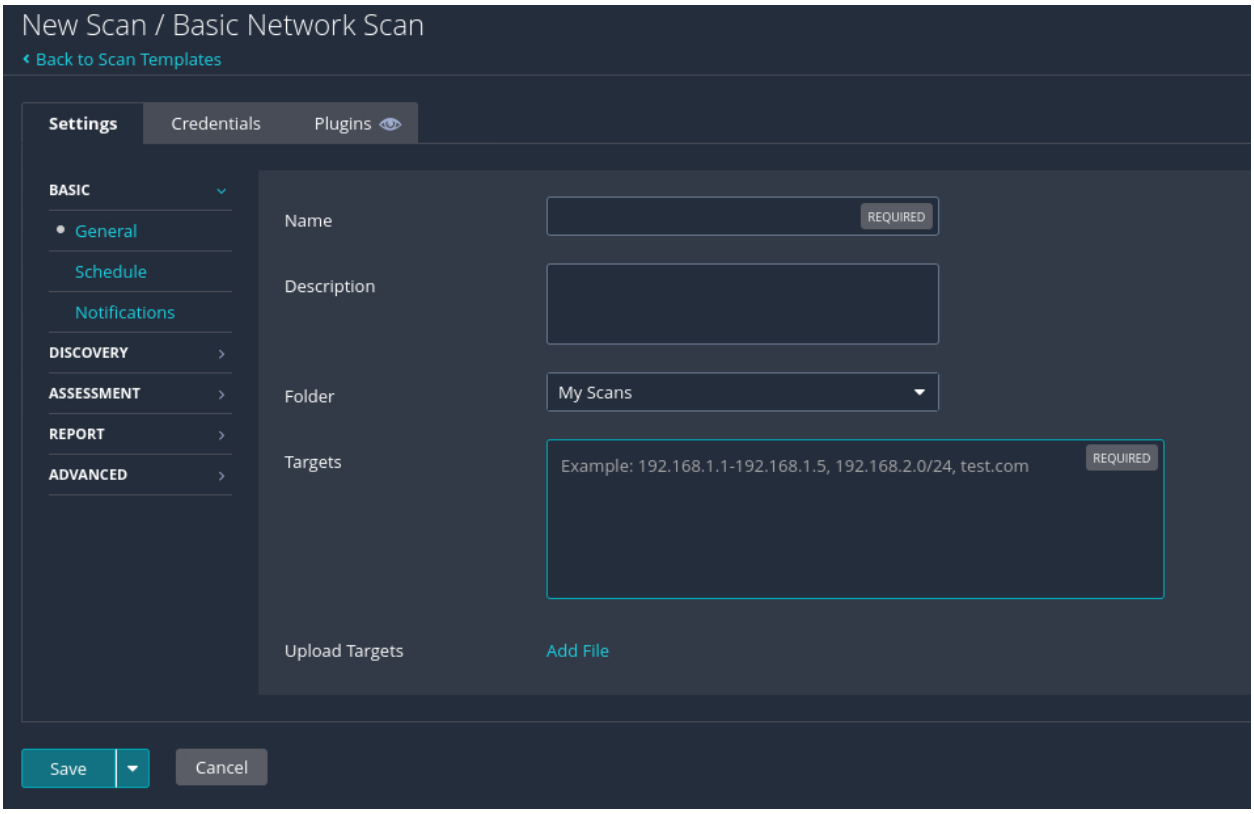
   cd ~/Downloads

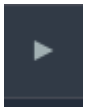
   sudo dpkg -i Nessus-10.11.1-debian10_amd64.deb

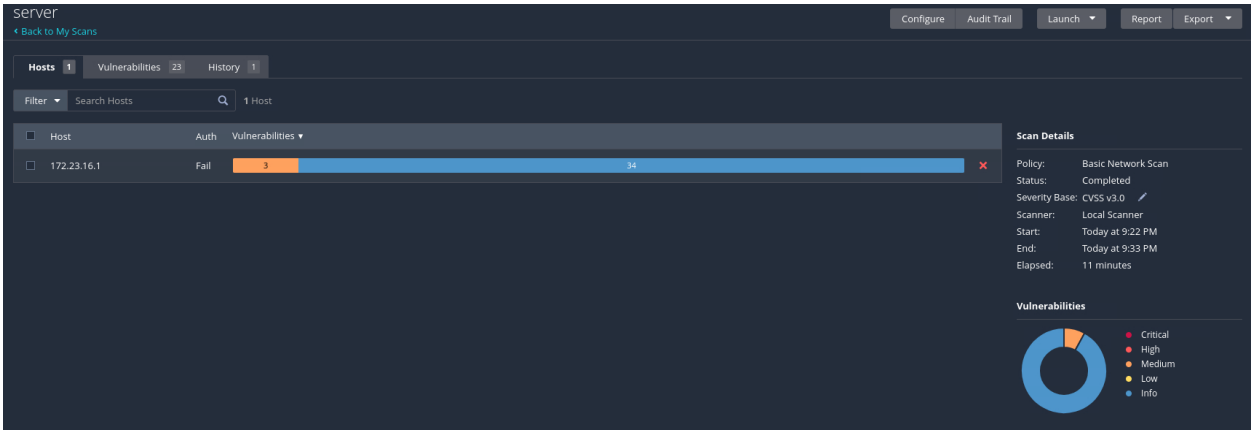
   sudo systemctl start nessusd

3. Відкрийте браузер, наберіть: https://localhost:8834 . Натисніть на advanced -> ignore certificate warning and enter website.
   На сторінці входу оберіть Nessus Essentials. Введіть свої ім’я, прізвище та email. У своєму поштовому клієнті, виберіть повідомлення від nessus та натисніть “verify your email address button”. Натисніть next на сторінці входу. Введіть ім’я адміна та пароль.
4. Панель Nessus відкриється. Зачекайте на завантаження та встановлення плагінів (це може тривати певний час 10-20 хв).

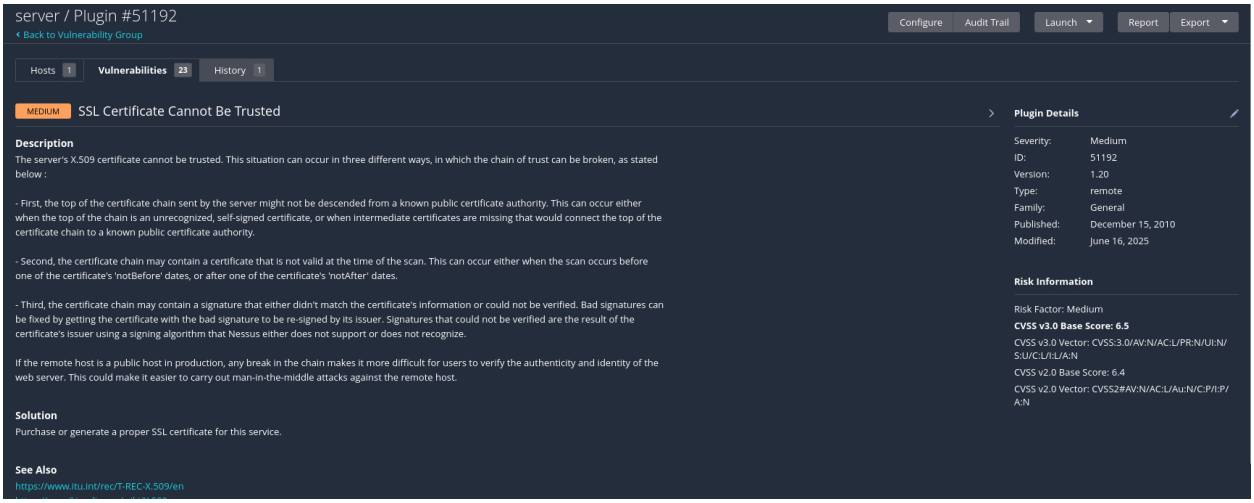
Після завантаження плагінів, оновіть сторінку. Натисніть “New scan” -> Basic network scan. Вкажіть назву та ip-адресу цілі в полі field. Натисніть кнопку save:



Натисніть кнопку  для запуску сканування, сканування буде тривати певний час, після завершення виберіть його в списку



Перейдіть до вкладки вразливостей(vulnerabilities), щоб відкрити список звітів щодо кожного просканованого елемента. Натисніть на назву у списку, щоб відкрити повний звіт про тип вразливості, фактор ризику тощо.



Автоматизоване тестування SQL-ін’єкцій (sqlmap)

Sqlmap: використовується для автоматизованого тестування SQL-ін’єкцій у JSON-API.

Використання:

1. Створіть файл login.req з наступним вмістом:

```
POST https://apiaddress:8443/api/auth/login HTTP/1.1
Host: apiaddress
Content-Type: application/json
User-Agent: curl/8.17.0
Accept: */*
Content-Length: 34
Connection: close

{"login":"test","password":"test"}
```

2. Відкрийте консоль:

   sqlmap -r ~/path_to_file/login.req --batch --level=1 --risk=1 --force-ssl

Якщо відповідь "all tested parameters do not appear to be injectable,", змінюйте level і risk на наступне число.

Можна змінювати:

- level від 1 до 5
- risk від 1 до 3

для кращих результатів.

Вибір параметрів: --technique=BEUSTQ (Boolean=B, Error=E, Union=U, Stacked=S, Time=T, Query=Q):

sqlmap -r ~/path_to_file/login.req –technique=T --batch --level=1 --risk=1 --force-ssl

Вказання БД(якщо відомо) db –dbms= для зменшення часу пошуку:

sqlmap -r login.req --dbms=mysql --batch --level=1 --risk=1 --force-ssl

Випадковий User-Agent для маскування:

sqlmap -r login.req –random-agent --batch --level=1 --risk=1 --force-ssl

Пауза між запитами для обходу внутрішнього захисту сервера:

sqlmap -r login.req –delay=1 --batch --level=1 --risk=1 --force-ssl

Тест з авторизацією:

sqlmap -r login.req –delay=1 --batch --force-ssl -H "Authorization: Bearer <token>"

Ручне тестування API за допомогою curl та Burp Suite

Тестування запитів з curl (подібно до postman)

Використання:

curl -k https://apiaddress:8443/api/auth/login -H "Content-Type: application/json" -d '{"login":"test","password":"test"}'

Логування всього трафіку для аналізу:

curl -k https://apiaddress/api/auth/login -H "Content-Type: application/json" -d '{"login":"test","password":"test"}' --trace-ascii ~/Downloads/trace.txt

Тест з авторизацією:

curl -vk https://apiaddress/api/dev/read \ -H "Authorization: Bearer YOUR_TOKEN" \ -H "Content-Type: application/json" \ -d '{"deviceId":1}'

Burp Suite (ручне тестування)

Burp Suite використовується як інструмент перехоплення та повтору HTTP(S)-запитів для ручного тестування автентифікації, авторизації та валідації вводу JSON-API.

1. Відкрийте burpsuite. Перейдіть на вкладку **Proxy → Proxy Settings**. Перевірте чи проксі(listener): **127.0.0.1:8080.** На вкладці Proxy, увімкніть **Intercept on.**

2. Curl використовується для генерації контрольованих API-запитів, які перехоплюються та аналізуються за допомогою Burp Suite Proxy. Введіть це в консолі:

   ```
   curl -vk -x http://127.0.0.1:8080 https://ipaddress:8443/api/auth/login \
     -H "Content-Type: application/json" \
     -d '{"login":"test","password":"test"}'
   ```

3. Перегляд запитів
   Відкрийте **Proxy → HTTP history.** Знайдіть захоплений пакет.
   Перевірте: HTTP method, Headers, JSON body, Response code.
4. Маніпуляція запитами (Repeater)

Режим Burp Repeater використовується для ручної модифікації json параметрів та перегляду відповідей від серверу.

Натисніть правою клавішею на запит в **HTTP history.** Виберіть **Send to Repeater.** Відкрийте **Repeater tab.** Модифікуйте запит.Натисніть **Send.** Аналізуйте змінену відповідь від серверу.

Burp Suite чеклист API-тестування

Автентифікація

 Видалити заголовок авторизації → очікувана помилка 401

 Змінити 1–2 символи в токені → очікувана помилка 401

 Повторний запит після розлогінення → токен відхилено

 Токен потрібен для всіх захищених кінцевих точок(endpoint)

 Токен не приймається в параметрах URL-адреси

Авторизація (IDOR) api/dev/read

 Зміна ID об’єктів (id, deviceId, userId)

 Доступ до ресурсів інших користувачів заборонено

 Великі ID значення відхиляються (999999)

 Від’ємні ID відхиляються (-1)

Валідація (JSON)

 Неправильні типи даних для devid тощо ("1", null, {}, [])

 Відсутні обов'язкові поля → очікувана помилка 400

 Додаткові поля ігноруються або відхиляються

 Глибокий/вкладений JSON відхиляється, якщо не підтримується

 Безпечна обробка символів Unicode / спеціальних символів

Масове призначення / Надмірні поля

 Неможливо встановити привілейовані поля (isAdmin, role)

{"deviceId":1,"isAdmin":true}

 Cannot override server-managed fields (userId)

{"deviceId":1,"role":"admin"}

 Extra JSON fields do not change behavior

{"deviceId":1,"userId":1}

SQL-ін’єкції (ручна перевірка)

 Лапки обробляються бзпечно (', ")

 Логічні ін’єкції не впливають (OR 1=1)

 Помилки SQL не виводяться у відповіді

 Без часових різниць (SLEEP, BENCHMARK)

 Однакова поведінка для всіх запитів

Приклади SQL ін’єкцій: https://github.com/Sourabh-Sahu/SQL-Injection/tree/main

Обробка помилок

 Неправильно сформований JSON оброблено коректно

 Немає повідомлень про помилки SQL

 Шляхи до файлів чи внутрішня інформація не були розкриті

 Коди помилок послідовні (400, 401, 403)

HTTP методи

 Дозволено лише цільові методи (POST)

 GET, PUT, DELETE відхиляються

 Заголовки перевизначення методу ігноруються

Кінцеві точки(Endpoint)

 Admin endpoints захищено

 Debug/test endpoints вимкнено

 Незадокументовані endpoints доступні

 Вгадування шляху не розкриває дані

Обмеження запитів / зловживання

 Обмеження кількості запитів для login

 Повторні запити не спричиняють уповільнення

 Неможливо застосувати bruteforce через API

Робота з токенами

 Термін дії токена закінчується після налаштованого часу

 Токен обмежений користувачем/сеансом

 Токен не можна використовувати повторно в різних ролях

 Токен не прийнято після зміни привілеїв

5. Автоматизоване тестування параметрів (Intruder)

Intruder = автоматичне повторення одного запиту з невеликими варіаціями (payloads).

Натисніть правою клавішею на запит в **HTTP history.** Натисніть **Send to Intruder.** Відкрийте вкладку Intruder**.**

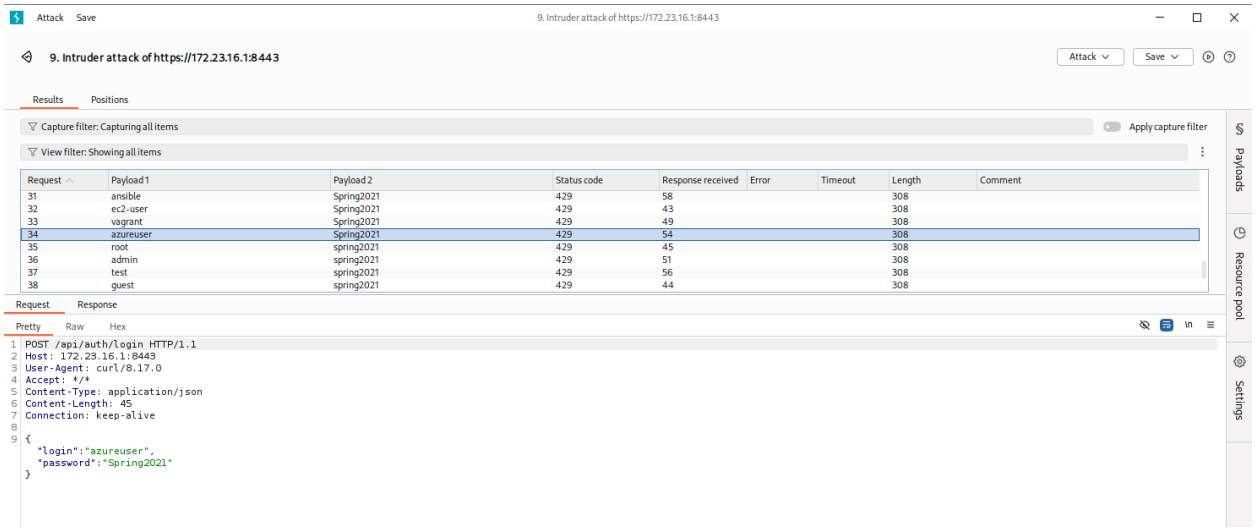
Виділіть потрібний параметр і натисніть **Add §**. Введіть список корисних навантажень(payloads) у вікні конфігурації, додавши їх вручну за допомогою кнопки Додати, кнопки Вставити, щоб вставити з буфера обміну, або кнопки Завантажити, щоб завантажити з файлу (txt). Потім натисніть **“Start attack on top”**.

Приклади тестування payload-ів:

Sniper attack(тестування одного параметру): перейдіть на https://github.com/danielmiessler/SecLists/blob/master/Usernames/top-

usernames-shortlist.txt. Збережіть дані в txt. Виділіть параметр входу ось так: `{"login":"§test§",`. Натисніть кнопку «Load» в конфігурації payload, перейдіть до вашого файлу topusernames.txt і натисніть кнопку «Відкрити». Значення імені користувача будуть відображені у списку. Натисніть кнопку «Start attack». Відкриється нове вікно.

Cluster bomb attack(перевіряє всі комбінації, ідеально підходить для кількох параметрів (username + password) для перевірки захисту від brute-force. Повторіть дії, які вказані в Sniper attack для першого параметру(login), потім виділіть параметр пароля ось так: `§test§`. Натисніть кнопку Load в конфігурації payload, перейдіть по цьому шляху: /usr/share/wordlists/ і виберіть fasttrack.txt, тоді натисніть Open. Щоб зробити атаку успішнішою та вразливішою до обмежень сервера щодо кількості запитів за секунду, перейдіть в Resource pool -> задайте значення Maximum concurrent requests = 1, і задайте delay between requests = 1000 milliseconds. Тоді натисніть Start attack кнопку. Відкриється нове вікно:



Request	Payload 1	Payload 2	Status code	Response received	Error	Timeout	Length	Comment
31	ansible	Spring2021	429	58			308	
32	ec2-user	Spring2021	429	43			308	
33	vagrant	Spring2021	429	49			308	
34	azureuser	Spring2021	429	54			308	
35	root	spring2021	429	45			308	
36	admin	spring2021	429	51			308	
37	test	spring2021	429	56			308	
38	guest	spring2021	429	44			308	

Натиснувши елемент у списку, ви можете побачити http-код запиту та http-код відповіді.

6. Аналіз токенів (**Sequencer**)

Sequencer(тестування на рандомність та стійкість).

Залогуйтесь в якості адміна в сервері у вкладці repeater, щоб зегенерувати токен. У відповіді, виділіть значення токена, правою клавішою миші -> send to sequencer. У вкладці sequencer, налаштуйте розташування токена, натиснувши на кнопку configure поруч з custom location field. У новому вікні знову виділіть токен і натисніть «ОК»::

Натисніть на “Start live capture”.

Навантажувальне тестування (Load / Stress)

- скільки користувачів витримує сервер

  Для тестування використовується Apache Bench (ab)

  Використання:

  ab -n 1000 -c 10 http://apiaddress

  -n 1000 → 1000 запитів

  -c 10 → 10 одночасних “користувачів”

  Відповідь:

  Requests per second: 120.45

Time per request: 83 ms

Failed requests: 0

Якщо Failed requests > 0 — сервер не витримує.

Потрібно робити поступове збільшення навантаження:

ab -n 5000 -c 50 http://ipaddress

ab -n 10000 -c 100 http://ipaddress

Сервер “падає”, коли:

- багато Failed requests
- відповіді стають дуже повільними
- з’являються 502 / 503 / 504
- сервер перестає відповідати

- **де «вузькі місця»**

  Показники, які вказують на проблему

  CPU

- Time per request росте
- сервер повільний, але не падає

  Пам’ять

- сервер падає раптово
- 500 помилки

Мережа

- таймаути
- нестабільні відповіді