

# Fundamentos de Análisis de tráfico de red

---

Network Forensics 2026

Análisis profesional de tráfico corporativo

# Quien soy

---

## Miguel Herrero Collantes

- ▶ Ingeniero técnico de telecomunicación - Sistemas electrónicos
  - ▶ Ingeniero de Telecomunicación
  - ▶ **Responsable de Seguridad de red** - EEAS (Bruselas) (2021-Actualidad)
  - ▶ **Analista de SOC** - Consejo de la UE (Bruselas) (2017-2021)
  - ▶ **Consultor de ciberseguridad** - Empresas financieras (2015-2019)
  - ▶ **Técnico de ciberseguridad** - Incibe (2011-2017)
  - ▶ **Ingeniero de Red 3G/4G** - Datatronics (2008-2011)
- 

Correo: mhercol[@]gmail[.]com

# Apéndice — Material de Referencia

---

Este documento contiene el material teórico de base del curso.

Está pensado como **consulta posterior** a las sesiones presenciales.

## Índice:

- A. Modelo TCP/IP y encapsulamiento · B. Sistemas de numeración
- C. Formato PCAP / pcapng · D. Herramientas de manipulación de PCAP
- E. Protocolos de capa de enlace y red

# Contenido del curso

---

## Sesión 1 (2h): Visibilidad, Captura y Protocolos Clave

- ▶ Arquitectura de captura: TAPs, SPAN, Zero Trust, Cloud
- ▶ PCAP, tcpdump, BPF y Wireshark
- ▶ Protocolos clave: ARP, TCP/IP, DNS, HTTP/S
- ▶ Lab 1 [LogiCorp]: Ana y la receta secreta (mensajería + extracción de archivo)
- ▶ Lab 2 [LogiCorp]: Ann Dercov - email personal desde la red corporativa

## Sesión 2 (2h): Forense de Red y Caza de Amenazas

- ▶ TLS 1.3: descifrado con SSLKEYLOGFILE
- ▶ Metodología forense, fuentes y técnicas de análisis
- ▶ Lab 3 [LogiCorp]: Reconocimiento de la red interna
- ▶ NetFlow, Full Packet Capture, Arkime
- ▶ Lab 4 [LogiCorp]: Infección de Stewie-PC

**Apéndice disponible:** TCP/IP, OSI, Hex/Binario, formato PCAP, Editcap, Mergecap, Capinfos

# Mapa del Curso

CASO INTEGRADOR: Empresa "LogiCorp" – Incidente en curso

# Caso Integrador — LogiCorp

## Escenario que seguiremos durante el curso:

### La empresa:

- ▶ **LogiCorp S.L.** — empresa logística, 300 empleados
- ▶ Red corporativa mixta: on-premise + AWS
- ▶ Sin SOC propio, solo un firewall perimetral

### El incidente:

- ▶ El usuario de **Stewie-PC** reporta que su equipo va lento
- ▶ IT detecta tráfico saliente inusual en el firewall
- ▶ **Nos llaman para investigar**

### Evidencias recibidas:

- ▶ **Evidencia01.pcap** — tráfico de la red WiFi corporativa
- ▶ **Evidencia02.pcap** — tráfico SMTP capturado en el gateway
- ▶ **Evidencia03.pcap** — captura del segmento de red interno
- ▶ **Evidencia04.pcap** — sesión de navegación del equipo infectado

### Fases de la investigación:

- ▶ ¿Cómo se exfiltró el activo crítico?
- ▶ ¿Se usó email para coordinar con el exterior?
- ▶ ¿Qué sistemas fueron reconocidos

# Arquitectura de Captura Corporativa

---

## Visibilidad Norte-Sur

- ▶ Tráfico que cruza el perímetro
- ▶ Firewalls, Proxies, Internet Gateway

## Visibilidad Este-Oeste

- ▶ Tráfico lateral entre servidores
- ▶ Microservicios
- ▶ **El más difícil de capturar**

## Acceso al Dato

### TAP (Test Access Point)

- ▶ [OK] Copia física
- ▶ [OK] Infalible
- ▶ [X] Costoso

### SPAN/Mirror Port

- ▶ [OK] Copia lógica
- ▶ [X] Puede perder paquetes
- ▶ [OK] Económico

### Cloud

- ▶ VPC Flow Logs
- ▶ Virtual TAPs

# El Desafío Zero Trust (ZT)

Cifrado Everywhere

mTLS y TLS 1.3 ocultan el payload incluso internamente

Microsegmentación

Tráfico lateral aislado, no pasa por Core Switch

Impacto Forense

PCAP inútil sin llaves de sesión

Estrategia: SSLKEYLOGFILE

Recolección de llaves en endpoints para descifrado

**Desafío 2025:** El 80% del tráfico interno corporativo está cifrado

# PCAP (Packet Capture)

## ¿Qué es PCAP?

- ▶ Packet **C**apture (captura de paquetes)
- ▶ Graba la actividad de red **completa** de las capas 2 a 7

## Formato más común: libpcap

- ▶ [OK] Open source
- ▶ [OK] Disponible en \*nix y Windows
- ▶ [OK] Librería en C
- ▶ [OK] Módulos en muchos lenguajes

### Usos principales:

- ▶ Investigación forense
- ▶ Debugging de red
- ▶ Análisis de malware
- ▶ Respuesta a incidentes
- ▶ Entrenamiento y educación

# Quién usa PCAP

---

## Investigadores

- ▶ Acceder a información en crudo
- ▶ Análisis profundo de protocolos

## Administradores de red

- ▶ Depurar problemas de red
- ▶ Optimización de rendimiento

## Analistas de seguridad

- ▶ Analizar actividad de malware
- ▶ Caracterizar amenazas

## Respuesta a incidentes

- ▶ Perseguir malware
- ▶ Recopilar evidencias
- ▶ Timeline reconstruction

# Tcpdump

---

## La herramienta de captura más usada

- ▶ [OK] Open Source
- ▶ [OK] Multiplataforma
- ▶ [OK] Basada en libpcap

### Características:

- ▶ Usa sintaxis **BPF** (Berkeley Packet Filter)
- ▶ Muestra detalles en terminal o guarda en pcap
- ▶ Lee de la red o de un pcap existente

```
# Captura básica  
tcpdump -i eth0  
  
# Guardar a archivo  
tcpdump -i eth0 -w captura.pcap  
  
# Con filtro BPF  
tcpdump -i eth0 'host 10.0.0.1'  
  
# Leer de archivo  
tcpdump -r captura.pcap
```

# Lectura/Escritura de pcaps

---

## Tcpdump

- ▶ Línea de comandos
- ▶ Captura desde interfaz de red

## Wireshark

- ▶ GUI completa
- ▶ Lee desde interfaz o archivo

## Tshark

- ▶ Wireshark en línea de comandos
- ▶ Scriptable y automatizable

## Scapy

- ▶ Herramienta en Python
- ▶ Leer, escribir y manipular paquetes

## Libtins

- ▶ Librería C++
- ▶ Más rápida que libpcap
- ▶ Para procesamiento masivo

# BPF (Berkeley Packet Filter)

## ¿Qué es BPF?

- ▶ Sintaxis para **filtrar paquetes** rápidamente
- ▶ Usada por **tcpdump** y **wireshark** (tshark)
- ▶ **Esencial** para analistas
- ▶ **Optimizada** por el kernel

### Ventaja:

Filtrado a nivel de kernel = Máximo rendimiento

```
# Ejemplo básico  
tcpdump -i eth0 -w archivo.pcap \  
'host 10.10.10.1 and tcp port 443'
```

# BPF - Sintaxis básica

---

## Modificadores:

### Type (Tipo)

- ▶ host - Dirección IP
- ▶ net - Red
- ▶ port - Puerto

### Dir (Dirección)

- ▶ src - Origen
- ▶ dst - Destino
- ▶ (por defecto: ambos)

### Proto (Protocolo)

- ▶ tcp, udp, icmp
- ▶ arp, ip, ip6

## Operadores booleanos:

- ▶ and (&&)
- ▶ or (||)
- ▶ not (!)
- ▶ () para agrupar

## Funciones:

- ▶ len - Número de bytes del paquete

```
# Ejemplos combinados
host 10.0.0.1 and tcp port 80

src net 192.168.0.0/16 and \
      dst port 443

tcp and (port 80 or port 443)
```

# BPF - Filtros por tamaño y flags

Filtrar por tamaño del paquete

- ▶ 'len <= 64' → Paquetes pequeños (SYN, RST, DNS)
- ▶ 'len > 1400' → Paquetes grandes (datos HTTP, exfiltración)

Filtrar UDP/IP de origen hacia direcciones privadas

- ▶ 'udp[8:2] > 10 && ip[16:4] <= 0xC0A80000'

Captura solo paquetes TCP con SYN (handshakes)

- ▶ 'tcp[13] & 2 != 0'

Paquetes con carga útil HTTP (método GET/POST)

- ▶ 'tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x47455420'

# BPF - Detección de amenazas

```
Captura paquetes con User-Agent sospechoso
▶ 'tcp[((tcp[12:1] & 0xf0) >> 2):11] = "curl"'  
  
Tráfico TLS con certificate exchange
▶ 'tcp[((tcp[12:1] & 0xf0) >> 2):1] = 0x16 && \
  tcp[((tcp[12:1] & 0xf0) >> 2)+5:1] = 0x0b'  
  
Paquetes con TTL bajo (possible IP spoofing)
▶ 'ip[8] < 10'  
  
Filtra RST inmediatos tras SYN (port scan)
▶ 'tcp[13] = 0x14 && len = 40'
```

# BPF - Filtros avanzados

---

Paquetes UDP con payload superior a 100 bytes

- ▶ 'udp && len > 100'

Paquetes ICMP con código de error

- ▶ 'icmp[0] = 3 || icmp[0] = 11'

Tráfico a puertos no estándar (evasión)

- ▶ 'tcp dst portrange 1024-65535 && \tcp[((tcp[12:1] & 0xf0) >> 2):4] = 0x504f5354'

Fragmentos IP (posible ataque de fragmentación)

- ▶ 'ip[6:2] & 0x1fff != 0'

# Wireshark - La herramienta esencial

---

## Wireshark

► Herramienta **principal** para análisis de red

► **Open source**

► GUI completa e intuitiva

## Características:

► [OK] Excelente parseo de protocolos

► [OK] Filtros potentes

► [OK] Búsqueda de cadenas de texto

► [OK] Exportación (archivos, objetos HTTP, descifrado)

- [OK] Estadísticas detalladas
- [OK] Datos expertos (Expert Info)
- [OK] Información sobre protocolos
- [OK] Análisis de conversaciones
- [OK] Seguimiento de streams (TCP, UDP, HTTP)

### Tip:

Dominar Wireshark es fundamental para cualquier analista de red

# Crear filtros en Wireshark

Opción 1: Barra de filtros — sintaxis directa con **autocompletear**

# Crear filtros en Wireshark

Opción 2: Botón **Expression** — selección guiada por protocolo y campo

# Crear filtros en Wireshark

Opción 3: Click derecho sobre un paquete o campo → *Apply as Filter*

# Crear filtros en Wireshark

---

## Filtros de lectura (Display Filters)

---

Al abrir un **PCAP** se puede aplicar un filtro de lectura:

- ▶ Solo aparecen los paquetes que cumplen las condiciones
- ▶ Útil para **reducir PCAPs grandes**
- ▶ No destruye datos — se puede quitar el filtro

### Diferencia clave:

**Capture Filter** = filtra durante la captura (BPF)

**Display Filter** = filtra la vista del archivo ya capturado

# Exportar objetos de Wireshark

---

File → Export Objects → HTTP

Wireshark puede **reconstituir y exportar** objetos

HTTP:

- ▶ Páginas HTML
- ▶ Imágenes descargadas
- ▶ Binarios / ejecutables
- ▶ Documentos (PDF, Office...)

Se guardan para análisis forense posterior

# LogiCorp — Apertura del PCAP

## En el caso de LogiCorp:

IT nos entrega el PCAP del firewall perimetral de las últimas 4 horas. Lo primero es orientarse sin perdernos en los miles de paquetes.

## Filtros iniciales aplicados:

- ▶ `dns` — aislamos todas las consultas de nombres
- ▶ `ip.src == 10.10.1.45` — solo tráfico de DESKTOP-MK3
- ▶ `tcp.port == 443 && ip.dst != 10.0.0.0/8` — HTTPS saliente

## Primer hallazgo:

Al aislar DNS, aparecen consultas repetidas al mismo dominio desconocido — exactamente cada 60 segundos.

Pasamos al módulo de protocolos para entender qué estamos viendo.

## Un poquito de protocolos...

“*Attackers bend and break protocols in order to smuggle covert data, sneak past firewalls, bypass authentication, and conduct widespread denial-of-service (DoS) attacks.*”

— Davidoff & Ham, 2012

”

## Capa de enlace IEEE 802.x — Trama

---

---

**Tamaño mínimo de trama:**  $14 + 46 + 4$  bytes (mecanismo anti-colisiones)

## Capa de enlace IEEE 802.x — Wireshark

---

# ARP — Address Resolution Protocol

---

## ¿Para qué sirve ARP?

- ▶ Resolución de dirección IP → MAC en el mismo segmento LAN
- ▶ Necesario antes de cualquier comunicación L2

## Flujo:

---

- ▶ A quiere hablar con 10.0.0.2 pero no sabe su MAC
- ▶ A emite un ARP Request (broadcast)
- ▶ 10.0.0.2 responde con su MAC (ARP Reply unicast)
- ▶ A almacena la entrada en su cache ARP

## ARP — En Wireshark

---

# Problemas de ARP

## ARP no es seguro:

- ▶ No valida autenticidad del emisor
- ▶ Cualquiera puede responder a cualquier petición ARP
- ▶ Todos los equipos actualizan su caché al recibir ARP

## ARP Spoofing / Poisoning:

- ▶ Un atacante envía ARP Replies falsos
- ▶ Redirige tráfico hacia sí mismo → **MITM**
- ▶ Permite capturar o modificar tráfico

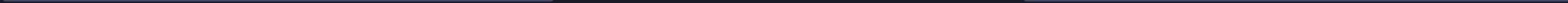
## Detección en Wireshark:

`arp.duplicate-address-detected` o múltiples MACs para la misma IP

# IPv4 vs IPv6

---

## IPv4

- ▶ Direccionamiento de **32 bits**
  - ▶ ~4.300 millones de direcciones
  - ▶ RFC 791
- 

## IPv6

- ▶ Direccionamiento de **128 bits**
- ▶ 340 undecillones de direcciones
- ▶ RFC 2460, RFC 4291

# Cabecera de IPv4

---

## Campos clave:

- ▶ **Versión**: 4 (IPv4) o 6 (IPv6) — otro valor → descarte
- ▶ **Protocol**: tipo de capa 4 encapsulada
  - ▷ ICMP: 1 | TCP: 6 | UDP: 17 (0x11)
- ▶ **TTL**: saltos máximos antes del descarte
  - ▷ Windows: 128 | Linux: 64 | Routers: 255
- ▶ **ECN**: notificación de congestión
  - ▷ 00 = No ECN | 01/10 = ECN-Aware | 11 = Congestión

# Cabecera IPv4 — Campos adicionales

---

## Tamaño y calidad de servicio:

- ▶ **IHL**: longitud de cabecera en palabras de 32 bits
  - ▷ Mínimo: 5 (20 bytes) | Máximo: 15 (60 bytes)
- ▶ **DSCP/ToS**: prioridad y clase de tráfico
  - ▷ VoIP, vídeo → valores elevados de DSCP
- ▶ **Total Length**: tamaño total del paquete (cabecera + datos)
  - ▷ Máximo: 65 535 bytes

## Fragmentación e integridad:

- ▶ **Identification**: ID común a todos los fragmentos del mismo datagrama
- ▶ **Flags**: control de fragmentación
  - ▷ DF (Don't Fragment) | MF (More Fragments)
- ▶ **Fragment Offset**: posición del fragmento (múltiplo de 8 bytes)
- ▶ **Header Checksum**: verificación de integridad solo de la cabecera
  - ▷ Recalculado en cada router (el TTL cambia)

# Fragmentación de IPv4

## ¿Cuándo ocurre?

- ▶ Paquete mayor que el **MTU** del enlace
  - ▷ Máx IP: 64KB | Ethernet: 1500 bytes

## Control de fragmentación:

- ▶ **ID**: mismo valor en todos los fragmentos
- ▶ **Offset**: posición del fragmento (múltiplo de 8)
- ▶ **Flag M**: "More Fragments" — activo en todos salvo el último

### Forense:

La fragmentación puede usarse para evadir IDS que solo inspeccionan el primer fragmento

# IPv6 — Ventajas

## Motivación:

- ▶ Direcciones IPv4 públicas agotadas desde ~2011
- ▶ NAT y redes privadas son parches temporales

## Mejoras técnicas:

- ▶ Enrutamiento más sencillo (sin broadcast)
- ▶ **Seguridad nativa** (IPSec integrado)
  - ▶ Cifrado del payload
  - ▶ Comprobación de integridad
  - ▶ Autenticación del origen

## Más mejoras:

- ▶ QoS mejorado (Flow Label)
- ▶ Payloads mayores (jumbogramas)
- ▶ Autoconfiguración SLAAC
- ▶ Sin checksums en cabecera

**Reto forense:** Mayor opacidad si no se tiene visibilidad sobre el enrutamiento IPv6 interno

# TCP vs UDP

---

## TCP

### Transmission Control Protocol

- ▶ **Confiable**
- ▶ Con secuenciación y reordenación
- ▶ Orientado a conexión (3-way handshake)
- ▶ Control de flujo y congestión
- ▶ Puertos 0–65535 | RFC 793

## UDP

### User Datagram Protocol

- ▶ **No confiable**
- ▶ Sin secuenciación
- ▶ No orientado a conexión
- ▶ Sin control de flujo
- ▶ Puertos 0–65535 | RFC 768

## TCP vs UDP — Cabeceras

---

# Establecimiento de conexión TCP (3-Way Handshake)

---

# Fin de conexión TCP (4-Way Teardown)

---

# UDP — Características

## Protocolo ligero y rápido:

- ▶ Sobrecarga mínima de cabecera (8 bytes)
- ▶ Sin control de flujo ni retransmisiones
- ▶ Sin garantía de orden de llegada

## Protocolos basados en UDP:

- ▶ **DNS** — resolución de nombres
- ▶ **NTP** — sincronización de tiempo
- ▶ **SNMP** — monitorización
- ▶ **DHCP** — configuración dinámica
- ▶ **QUIC/HTTP3** — web moderno

## Cabecera UDP (8 bytes):

Puerto origen | Puerto destino  
Longitud total | Checksum

Diseñados en una era sin modelo de seguridad → son vectores de ataque habituales

## UDP — En Wireshark

---

**Tamaño mínimo:** 8 bytes (solo cabecera) — si vale 0 = jumbograma

# ICMP — Internet Control Message Protocol

## Propósito:

- ▶ Reportar errores **no temporales** de la red
- ▶ Intercambiar información de control simple

## Casos de uso:

- ▶ Fragmentación necesaria pero DF=1
- ▶ Puerto inalcanzable (destino no responde)
- ▶ **Ping** (Echo Request / Echo Reply)
- ▶ TTL Exceeded (traceroute)

## Habitual bloquearlo en redes corporativas:

- ▶ Dificulta el descubrimiento de la red
- ▶ Anula PMTUD → puede generar problemas de red

## Forense:

Los mensajes de error ICMP **incluyen parte del paquete original** que causó el error → revelan información de sesiones internas

# ICMP Echo Request / Reply (PING)

## Estructura:

- ▶ **Mismo identificador** en request y reply
- ▶ Diferentes **números de secuencia** por envío
- ▶ Permite controlar paquetes perdidos y latencia

## Uso malicioso:

- ▶ **ICMP Tunneling** — exfiltración de datos en el payload ICMP
- ▶ Reconocimiento de red (ping sweep)
- ▶ Fragmentación maliciosa (Ping of Death)

## Detección en Wireshark:

Filtro: `icmp.type == 8` (request)

Payload ICMP anormalmente grande = sospechoso

# DNS — Domain Name System

---

## ¿Qué hace DNS?

- ▶ Traduce nombres de dominio en direcciones IP
- ▶ Jerarquía de servidores involucrados:
  - ▷ **Resolutores** (stub + recursive)
  - ▷ **Raíz** (13 grupos de root servers)
  - ▷ **TLD** (.com, .es, .org...)
  - ▷ **Autoritativos** (zona del dominio)

## Secciones de la respuesta DNS:

- ▶ **Question:** el nombre consultado
- ▶ **Answer:** registros que responden la consulta
- ▶ **Authority:** servidores autoritativos del dominio
- ▶ **Additional:** IPs de los autoritativos

## Forense DNS:

DNS es el protocolo de exfiltración y C2 más usado — todo el tráfico interno pasa por el DNS corporativo

# DNS Records — Tipos

---

Tipo	Función	Relevancia Forense
A	IPv4 del dominio	Resolución C2, phishing
AAAA	IPv6 del dominio	C2 sobre IPv6
CNAME	Alias	Redirección encubierta
MX	Servidor de correo	SPAM, phishing
TXT	Texto libre	Exfiltración, verificación
PTR	Reverse DNS (IP → nombre)	Atribución
NS	Servidor autoritativo	Takeover de dominio

# DNS — Variantes de transporte

---

## DNS clásico (UDP/53):

- ▶ Sin cifrado — visible en la red
- ▶ Fácil de analizar en Wireshark
- ▶ Fácil de bloquear o interceptar

## DNS sobre TLS (DoT / puerto 853):

- ▶ Tráfico cifrado TCP
- ▶ Más fácil de bloquear (puerto único)

## DNS sobre HTTPS (DoH / puerto 443):

- ▶ Mezclado con tráfico HTTPS normal
- ▶ **Difícil de bloquear sin rompimiento TLS**
- ▶ Usado por malware moderno para evasión

### Reto forense 2025:

Malware usa DoH (Cloudflare, Google) para ocultar resoluciones de C2

# LogiCorp — Anomalía DNS detectada

En el caso de LogiCorp:

Filtramos dns y ordenamos por nombre de dominio. Aparece upd4te-cdn.net con 240 consultas en 4 horas.

Señales de alarma:

- ▶ Intervalo exacto de 60 segundos → **beaconing**
- ▶ Dominio registrado hace 3 días → sospechoso
- ▶ Sin tráfico web previo a ese dominio → no es legítimo
- ▶ Responde siempre a la misma IP:  
185.220.101.12

Filtro Wireshark:

dns.qry.name contains "upd4te"

→ 240 resultados, todos desde DESKTOP-MK3

Regla práctica: si un dominio aparece con **periodicidad fija**, es C2 hasta que se demuestre lo contrario

# HTTP/S — HyperText Transfer Protocol

## HTTP (puerto 80)

- ▶ Protocolo de texto claro
- ▶ Uso muy reducido hoy en día
- ▶ Visible directamente en Wireshark

## HTTPS (puerto 443)

- ▶ HTTP sobre TLS/SSL — cifrado end-to-end
- ▶ Requiere interceptación mediante proxy o `SSLKEYLOGFILE`
- ▶ 95%+ del tráfico web actual

**Vector muy común y difícil de bloquear:**

- ▶ Navegación web legítima
- ▶ APIs REST
- ▶ Delivery de malware
- ▶ Canales C2
- ▶ Exfiltración de datos

Casi todos los campos HTTP son **manipulables** por el atacante

## HTTP/S — Códigos de respuesta

Rango	Significado	Ejemplo forense
1xx	Informativo	Poco común, ignorable
2xx	Éxito	200 OK — contenido servido
3xx	Redirección	302 → redirección a C2
4xx	Error cliente	404 fichero no existe, 403 acceso denegado
5xx	Error servidor	500 = posible explotación

**Truco forense:** Una ráfaga de 404 desde una IP = reconocimiento (fuzzing de rutas)

Una serie de 200 hacia URLs aleatorias = posible DGA / C2

# HTTP — Métodos

---

## Métodos comunes:

- ▶ **GET** — solicitar recurso (sin body)
- ▶ **POST** — enviar datos al servidor
- ▶ **PUT** — subir/actualizar recurso
- ▶ **DELETE** — eliminar recurso
- ▶ **HEAD** — solo cabeceras, sin cuerpo

## Relevancia forense:

- ▶ POST con body grande → posible exfiltración
- ▶ PUT/DELETE inusuales → posible webshell
- ▶ User-Agent sospechoso → herramienta de ataque
- ▶ Muchos GET rápidos → fuzzing / scaneo

# Descifrado de TLS 1.3 en Wireshark

## El problema con TLS 1.3:

- ▶ TLS 1.2 → se podía usar la clave privada del servidor
- ▶ TLS 1.3 → **Perfect Forward Secrecy obligatorio**
  - ▶ Cada sesión tiene claves efímeras distintas
  - ▶ La clave privada ya no descifra nada

## Solución: **SSLKEYLOGFILE**

- ▶ El navegador/app exporta las *session keys*
- ▶ Wireshark las usa para descifrar en tiempo real

```
# Linux / Mac  
export SSLKEYLOGFILE=~/tls_keys.log  
  
# Windows (permanente)  
setx SSLKEYLOGFILE C:\tls_keys.log  
  
# Wireshark:  
# Edit → Preferences → Protocols  
#   → TLS → Pre-Master-Secret log file
```

Solo funciona si **controlas el endpoint** que genera el tráfico



# Análisis Forense de Red — Definición

## ¿Qué es?

- ▶ El arte de **reunir evidencias** digitales en la escena de un crimen de red
- ▶ Monitorización y análisis del tráfico para:
  - ▶ Reunir información e indicios
  - ▶ Obtener pruebas con  **validez legal**
  - ▶ Detectar intrusos y actividad maliciosa

## Características:

- ▶ Maneja información **dinámica y muy volátil**
- ▶ Fuertemente dependiente de la **calidad y cantidad** de datos disponibles
- ▶ Requiere formación técnica **y** conocimiento del entorno

## Objetivo final:

Reconstruir las acciones del atacante con suficiente detalle para:

- ▶ Entender el vector de ataque
- ▶ Determinar el alcance del daño
- ▶ Atribuir la actividad
- ▶ Sustentar una acusación legal

# Marco Legal — ¿Puedo capturar este tráfico?

---

## Principio fundamental

Capturar tráfico de red **sin autorización** es ilegal en la mayoría de jurisdicciones, incluso siendo el administrador de la red

## ¿Cuándo es legal?

- ▶ Eres el **propietario de la red** o actúas con mandato explícito
- ▶ Existe una **política de uso aceptable** firmada por los empleados
- ▶ En España: **autorización escrita** de la dirección antes de capturar
- ▶ En entornos cloud: verificar los **términos del proveedor**

## Normativa clave (España / UE)

- ▶ **GDPR / RGPD** — el tráfico capturado puede contener datos personales
  - ▷ Obligación de minimización de datos
  - ▷ Retención limitada al tiempo necesario
- ▶ **Ley Orgánica 3/2018 (LOPDGDD)**
  - ▷ Complementa el GDPR en España
- ▶ **LECrim Art. 588 bis**
  - ▷ Intervención de comunicaciones: requiere autorización judicial en investigaciones penales

# Marco Legal — Cadena de Custodia

## ¿Por qué importa la cadena de custodia?

- ▶ Una evidencia sin cadena de custodia **no es admisible en juicio**
- ▶ El atacante puede alegar que los datos fueron alterados
- ▶ En incidentes graves, el caso judicial depende de la evidencia digital

## Elementos mínimos al documentar un PCAP:

- ▶ Hash del fichero (`sha256sum captura.pcap`)
- ▶ Fecha, hora y zona horaria de la captura
- ▶ Interfaz y sistema donde se capturó
- ▶ Quién lo recogió y quién ha tenido acceso
- ▶ Herramienta utilizada (versión incluida)

### Regla práctica:

Antes de tocar cualquier evidencia, haz un hash y guárdalo en un lugar separado. Es el equivalente digital de los guantes en una escena del crimen.

### En el caso de LogiCorp:

Antes de abrir el PCAP, documentamos: hash SHA-256, hora de descarga, nombre del analista. El fichero original queda en modo solo lectura.

# Análisis Forense de Red — Las 6 Preguntas

---

El forense de red examina el tráfico que ha atravesado la red y lo correlaciona con actividad anómala para responder:

Pregunta	Ejemplo
Quién	IP origen, MAC, usuario autenticado
Qué	Protocolo, payload, acción realizada
Dónde	IP/hostname destino, segmento de red
Cuándo	Timestamp, duración, frecuencia
Cómo	Técnica de ataque, herramienta usada
Por qué	Motivación inferida, objetivo

# Componentes del Análisis Forense de Red

## 1. Captura

- ▶ TAPs / SPAN ports
- ▶ Sensors en distintos segmentos
- ▶ Logs de dispositivos
- ▶ Flujos (NetFlow/IPFIX)

└ Sin captura no hay análisis

## 2. Indexación

- ▶ Organización temporal
- ▶ Extracción de metadatos
- ▶ Correlación entre fuentes
- ▶ Sistemas: Arkime, Elastic, Splunk

└ Permite buscar en TB de datos en segundos

## 3. Análisis

- ▶ Filtrado y segmentación
- ▶ Reconstrucción de sesiones
- ▶ Detección de patrones
- ▶ Correlación con IOCs

└ La inteligencia del analista es el factor clave

# Usos del Análisis Forense de Red

---

## Seguridad

- ▶ Detección de intrusos
- ▶ Identificar y definir firmas de tráfico malicioso
- ▶ Descubrimiento pasivo de equipos y servicios
- ▶ Examen forense post-incidente
- ▶ Tráfico como evidencia legal
- ▶ Auditoría de reglas de Firewall
- ▶ Validación de controles de acceso

## Troubleshooting

- ▶ Errores de configuración en dispositivos
- ▶ Diagnóstico de latencia y pérdida de paquetes
- ▶ Errores de red y de servicio

## Optimización

- ▶ Análisis del uso de ancho de banda
- ▶ Evaluación del tamaño de paquetes
- ▶ Tiempo de respuesta entre redes

## Análisis de aplicaciones

- ▶ Inventario de protocolos y puertos
- ▶ Validación de aplicaciones de seguridad

# Arquitectura de Red Corporativa Típica

## Empresa pequeña:

- ▶ LAN + servidores internos
- ▶ 1-2 firewalls para segmentación
- ▶ Segmento WiFi separado
- ▶ Conexión Internet directa

## A medida que crece:

- ▶ Varios edificios con Fibra Oscura privada
- ▶ Oficinas remotas (SD-WAN o VPN)
- ▶ Teletrabajadores
- ▶ Entornos de administración privilegiados
- ▶ Servidores en la nube (AWS/Azure/GCP)

## Implicaciones forenses:

Cuanto mayor es la red, más complejo el análisis:

- ▶ Múltiples puntos de captura necesarios
- ▶ Tráfico cifrado en todos los segmentos
- ▶ Visibilidad Este-Oeste limitada
- ▶ NAT y proxies ocultan IPs reales

# Información de Red Disponible

---

En un análisis forense de red, los orígenes de información posibles son:

- ▶ **Paquetes completos (Full Packet Capture)**

- ▷ Máxima granularidad, alto coste de almacenamiento

- ▶ **Flujos de información (NetFlow/IPFIX)**

- ▷ Solo metadatos, pero escalable

- ▶ **Combinación PCAP + flows**

- ▷ Lo ideal en entornos empresariales

- ▶ **Solo logs de dispositivos**

- ▷ Limitado pero útil para correlación

- ▶ **Sin información**

- ▷ El escenario más común en incidentes

# Paquetes Completos vs Sesiones

---

## Full Packet Capture (FPC)

Todo el contenido de la red: cabeceras + payload completo

## Información de Sesiones

Un intercambio temporal entre dos o más equipos — metadatos de la conversación

# Retos de la Recolección de Red

---

## Retos técnicos:

- ▶ **Periodo de retención corto** — el almacenamiento es caro
- ▶ ¿Tienes FPC de los eventos de interés?
- ▶ ¿Recolección en toda la red o solo en un segmento?
- ▶ **Tráfico cifrado** — TLS 1.3 oculta el payload
- ▶ Túneles y puertos no estándar (evasión)

## Retos organizativos:

- ▶ Atribución de **NAT** — múltiples hosts detrás de una IP pública
- ▶ Atribución de **DHCP** — IPs dinámicas cambian de equipo
- ▶ Riesgos legales del almacenamiento de tráfico privado

### El mayor reto de 2025:

El 80%+ del tráfico corporativo está cifrado — capturar no es suficiente

# Conocimientos Necesarios

---

## Técnicos:

- ▶ Networking — modelo TCP/IP, protocolos
- ▶ Protocolos de red y de aplicación más comunes
- ▶ Herramientas: Wireshark, tcpdump, tshark, Arkime
- ▶ Productos y aplicaciones de seguridad
- ▶ Amenazas y ataques típicos de red
- ▶ Scripting para automatización (Python, bash)

## Contextuales:

- ▶ Entorno de la empresa:
  - ▶ Arquitectura de red y segmentación
  - ▶ IPs de activos clave (FW, Proxy, DC, DNS...)
  - ▶ Patrones de tráfico normal (baseline)
  - ▶ Horarios y usuarios del sistema

**Sin conocer qué es normal, no se puede detectar lo anómalo**

# Metodología de Investigación Forense

## La metodología depende de:

- ▶ El **tipo de información disponible**
- ▶ El **tipo de incidente** (malware, insider, exfil...)
- ▶ Los **detalles específicos** del caso

## Propiedades:

- ▶ Es un **proceso iterativo** — no lineal
- ▶ No existe una fórmula única
- ▶ Los IOC descubiertos redirigen el análisis
- ▶ La hipótesis guía la búsqueda

## Ciclo básico:

1. **Alerta / Hipótesis** — punto de partida
2. **Búsqueda** — filtros, estadísticas, correlación
3. **Hallazgo** — nuevo IOC o artefacto
4. **Ampliar** — buscar más evidencias relacionadas
5. **Documentar** — cadena de custodia
6. **Repetir** hasta completar el timeline

# Metodología — Framework de Threat Hunting

## 1. HIPÓTESIS

¿Qué comportamiento anómalo busco?

→ Ejemplo: Beaconsing C2 cada 60 segundos hacia IP externa

## 2. BÚSQUEDA

Aplicar filtros y análisis estadístico

→ Ejemplo: dns.qry.name + análisis de intervalos temporales

## 3. VALIDACIÓN

¿Es malicioso o es tráfico legítimo?

→ Ejemplo: Correlación con VirusTotal, ThreatFox, contexto IOCs

## 4. REMEDIACIÓN

Bloquear, aislar, documentar el incidente

→ Ejemplo: Regla de firewall + ticket de incidente + notificación

## 5. RETROSCENARIOS

¿Cuándo comenzó realmente? ¿Afectó a otros hosts?

→ Buscar en datos históricos con los nuevos IOCs

# Fases de un Ataque — Desde la Perspectiva Forense

---

## Pre-explotación

### ► Reconocimiento

- ▷ DNS queries masivas, escaneo de puertos, fingerprinting
- ▷ Indicadores: muchos SYN sin completar handshake

## Explotación

### ► Delivery — descarga del payload

- ▷ HTTP/S con archivo .exe/.dll/.zip sospechoso

### ► Explotación del servicio vulnerable

- ▷ Payloads anómalos, tráfico malformado

## Post-explotación

### ► C2 / Mantener acceso

- ▷ Beaconing periódico, DNS tunneling, HTTPS a IPs sospechosas

### ► Elevación de privilegios

- ▷ Tráfico Kerberoasting, Pass-the-Hash

### ► Movimiento lateral

- ▷ SMB, RDP, WMI entre hosts internos

### ► Exfiltración

- ▷ Upload de datos, DNS tunneling, ICMP tunneling

# Técnicas de Análisis de Paquetes

## Búsqueda de cadenas

- ▶ Identificar paquetes que contengan valores específicos en el payload
- ▶ Útil para buscar credenciales, user-agents, URLs

```
# Wireshark  
frame contains "password"  
http.user_agent contains "curl"
```

## Filtrado de paquetes

- ▶ Separar paquetes según metadatos (IP, puerto, protocolo, flags)
- ▶ Reduce el espacio de búsqueda

```
# BPF en tcpdump  
tcp and host 10.0.0.1 and port 443
```

## Parseo de campos de protocolo

- ▶ Extraer contenido específico de campos del protocolo
- ▶ Analizar cabeceras HTTP, DNS records, flags TCP

```
# tshark - extraer campos  
tshark -r cap.pcap \  
-T fields \  
-e ip.src -e dns.qry.name
```

## Análisis estadístico

- ▶ Detectar patrones de beaconing por intervalos
- ▶ Top talkers / top destinations
- ▶ Distribución de tamaños de paquetes

# Técnicas Avanzadas de Análisis

---

## Reconstrucción de sesiones

- ▶ Follow TCP/UDP/HTTP Stream en Wireshark
- ▶ Reensamblar archivos transferidos
- ▶ Recuperar páginas web visitadas
- ▶ Exportar objetos HTTP / SMB

## Análisis de comportamiento

- ▶ Comparar con el **baseline** de la red
- ▶ Detectar conexiones fuera de horario
- ▶ Volúmenes de tráfico anómalos
- ▶ Nuevos hosts o protocolos no vistos antes

## Correlación de fuentes

- ▶ PCAP + logs de firewall + logs de proxy
- ▶ PCAP + logs de autenticación (AD/LDAP)
- ▶ PCAP + EDR (proceso que generó la conexión)
- ▶ PCAP + inteligencia de amenazas (IOCs)

### Regla de oro:

El PCAP es el "ground truth" — cuando el SIEM duda, el PCAP tiene la respuesta

**Infraestructura como fuente de evidencias**

**Cada dispositivo de red es un testigo potencial**

# Fuentes — WiFi y Switches

---

## Redes WiFi (WLAN)

- ▶ Información normalmente cifrada (WPA2/WPA3)
- ▶ \*\*Frames de gestión y control no suelen ir cifradas:\*\*
  - ▷ APs anuncian SSID, presencia y capacidades
  - ▷ MACs de equipos autenticados
  - ▷ Análisis de volumen de tráfico
- ▶ En WPA2 con PSK: se puede descifrar con la clave

## Switches (Capa 2)

- ▶ Interconectan segmentos de red locales
- ▶ \*\*Tabla CAM (Content Addressable Memory):\*\*
  - ▷ Mapea puertos físicos → MACs
  - ▷ Permite ubicar físicamente un dispositivo
  - ▷ Clave para localizar un equipo comprometido

```
show mac address-table | include [MAC]
```

# Fuentes — Routers

## Función:

- ▶ Conectan y encaminan tráfico entre diferentes redes
- ▶ Permiten comunicación MAN/WAN/LAN

## Información forense:

- ▶ **Tablas de enrutamiento** — el path de las comunicaciones
- ▶ **Filtrado de paquetes** — ACLs aplicadas
- ▶ **Logs de acceso** — eventos de conexión
- ▶ **Información de flujos** — NetFlow/IPFIX

Los routers son los **IDS más desplegados** (y más rudimentarios):

- ▶ Todos tienen ACLs
- ▶ Muchos tienen NetFlow habilitado
- ▶ Los logs suelen ir a un SIEM o syslog

Los logs del router pueden ser la única fuente de evidencia cuando no hay PCAP

# Fuentes — DHCP y DNS

---

## Servidores DHCP

- ▶ Asignan IP dinámicamente a equipos de la LAN
- ▶ \*\*La investigación suele comenzar con una IP\*\*
- ▶ Los logs DHCP contienen:
  - ▷ Timestamp de la petición
  - ▷ IP asignada y duración del lease
  - ▷ MAC del equipo solicitante
  - ▷ Hostname del solicitante

IP → MAC → hostname → usuario (vía AD)

## Servidores DNS

- ▶ Resuelven nombres para todos los equipos internos
- ▶ \*\*Log de peticiones recibidas:\*\*
  - ▷ Intentos de conexión interior → exterior
  - ▷ Timestamps de cada resolución
  - ▷ Permite crear la \*\*timeline\*\* de actividad sospechosa

Muchos ataques (C2, exfiltración) se delatan en los logs DNS antes que en cualquier otra fuente

# Fuentes — Autenticación y NIDS

---

## Servidores de Autenticación (AD/LDAP/RADIUS)

- ▶ Servicios de autenticación centralizada
- ▶ Provisionamiento y auditoría de cuentas
- ▶ \*\*Información forense:\*\*
  - ▷ Intentos fallidos → fuerza bruta
  - ▷ Éxitos en horas sospechosas
  - ▷ Localizaciones no habituales para el usuario
  - ▷ Cambios de privilegios inesperados

## NIDS/NIPS

- ▶ Monitorizan tráfico en tiempo real
- ▶ Detectan y alertan de eventos sospechosos
- ▶ \*\*Información forense:\*\*
  - ▷ Actividades sospechosas en curso
  - ▷ Tráfico hacia C2 conocidos
  - ▷ Fugas de información
  - ▷ Recuperación de contenido completo (en algunos casos)
  - ▷ Normalmente: IP origen/destino, puertos, timestamp

# Fuentes — Firewalls y Proxies

---

## Firewalls (NGFW)

- ▶ Inspección con tres acciones: permitir, descartar, registrar
- ▶ Basados en IP, puerto, protocolo y payload
- ▶ \*\*Información forense:\*\*
  - ▷ Log granular de tráfico permitido y denegado
  - ▷ Logs de cambios de configuración
  - ▷ Alertas IPS integradas
  - ▷ Identificación de aplicaciones (L7)

## Proxies Web

- ▶ Mejoran rendimiento mediante caché
- ▶ Registran, inspeccionan y filtran tráfico web
- ▶ \*\*Información forense:\*\*
  - ▷ Logs granulares de navegación (larga retención)
  - ▷ Perfiles de navegación por IP / usuario
  - ▷ Detección de phishing exitoso
  - ▷ Identificación de malware web
  - ▷ Contenido cacheado (lo que vio el usuario)

# Fuentes — Servidores de Aplicación y Logs Centralizados

---

## Servidores de Aplicación

- ▶ Bases de datos
- ▶ Servidores Web
- ▶ Servidores de correo (SMTP/IMAP)
- ▶ Servidores de mensajería (IM)
- ▶ Servidores VoIP

Guardan logs de sus aplicaciones — **esenciales** para descifrar qué ocurrió realmente a nivel de aplicación

## SIEM / Logs Centralizados

- ▶ Combina logs de muchas fuentes
- ▶ Correlación, análisis y fechado automático
- ▶ **\*\*Valor forense:\*\***
- ▶ Diseñado para identificar y responder a incidentes
- ▶ Salva la información si un servidor es comprometido
- ▶ Retiene datos durante más tiempo que los dispositivos
- ▶ Análisis forense y visualización temporal

# Pegasus — Caso Práctico de Análisis Forense

## La cadena de explotación:

- ▶ iPhone recibe SMS con un **.gif** a través de iMessage
- ▶ El archivo **no es un GIF** — es un PDF con extensión falsa
- ▶ imagemIO intenta parsear el PDF
- ▶ El PDF usa formato JBIG2 con operadores AND/OR/XOR/XNOR
- ▶ Vulnerabilidad de **Buffer Overflow** — acceso a memoria fuera del proceso
- ▶ El PDF es un script con +70.000 comandos lógicos
- ▶ Crea una arquitectura funcional equivalente a JS
- ▶ **Zero-click** — sin interacción del usuario

## Lección forense:

El payload malicioso estaba en el PDF — un análisis de la extensión del fichero no habría detectado nada

Análisis práctico de capturas reales

De la teoría a la investigación activa

# Errores Comunes en Análisis de Red

## No confiar ciegamente en timestamps:

NTP drift puede causar desfases entre equipos

**Solución:** Verificar: `ntpq -p` o fuente autoritativa

## No validar integridad del PCAP:

PCAP corrupto = análisis inválido

**Solución:** Siempre: `capinfos archivo.pcap`

## Ignorar MTU y fragmentación:

Payloads >1500 bytes se fragmentan

**Solución:** `ip.flags.mf == 1` o `tcp.analysis.retransmission`

## Buscar contraseñas sin considerar cifrado:

95% del tráfico web es HTTPS hoy en día

**Solución:** Busca en HTTP, FTP, Telnet, SMTP (legacy)

## No documentar la cadena de custodia:

PCAP sin hash = inadmisible en juicio

**Solución:** `md5sum *.pcap > checksums.txt` al capturar

## Analizar sin conocer la red:

¿Qué es normal? ¿Cuánto DNS es esperado?

**Solución:** Establece un **baseline** antes del incidente

# Lab 1 — LogiCorp: Fuga de la receta secreta

---

## Escenario:

IT de LogiCorp detecta actividad sospechosa en la red WiFi corporativa. Un portátil externo apareció brevemente en el parking y se conectó a la red.

El equipo de **Ana** ( **192.168.1.158** ), empleada con acceso al activo más crítico de la empresa — la **receta secreta** — intercambió mensajes instantáneos con ese portátil y transfirió un archivo. El portátil desapareció minutos después.

**PCAP:** Evidencia01.pcap

# Lab 1 — Buscamos Respuestas

## Preguntas a responder:

- ▶ ¿Con quién estaba hablando Ana?
- ▶ ¿Cuál es el primer mensaje de la conversación?
- ▶ ¿Cuál es el nombre del archivo que transfirió?
- ▶ ¿Puedes recuperar el documento intercambiado?

## Pistas:

- ▶ IP de Ana: 192.168.1.158
- ▶ Buscar protocolo de mensajería instantánea (IM)
- ▶ Seguir los streams TCP desde esa IP

# Lab 1 — Cómo encontrar el tráfico AIM

## Filtrar el protocolo:

- ▶ AIM usa el puerto **5190/TCP**
- ▶ Filtro: `tcp.port == 5190`
- ▶ Follow TCP Stream → mensajes en texto claro

## Lo que aparece en el stream:

- ▶ Nombre de usuario (`Sec558user1`) y los mensajes de chat
- ▶ El nombre del fichero en la cabecera OFT2 (*Oscar File Transfer*)
- ▶ El fichero completo (PK... = ZIP/DOCX) para extraerlo

## ¿Cuál es el primer mensaje?

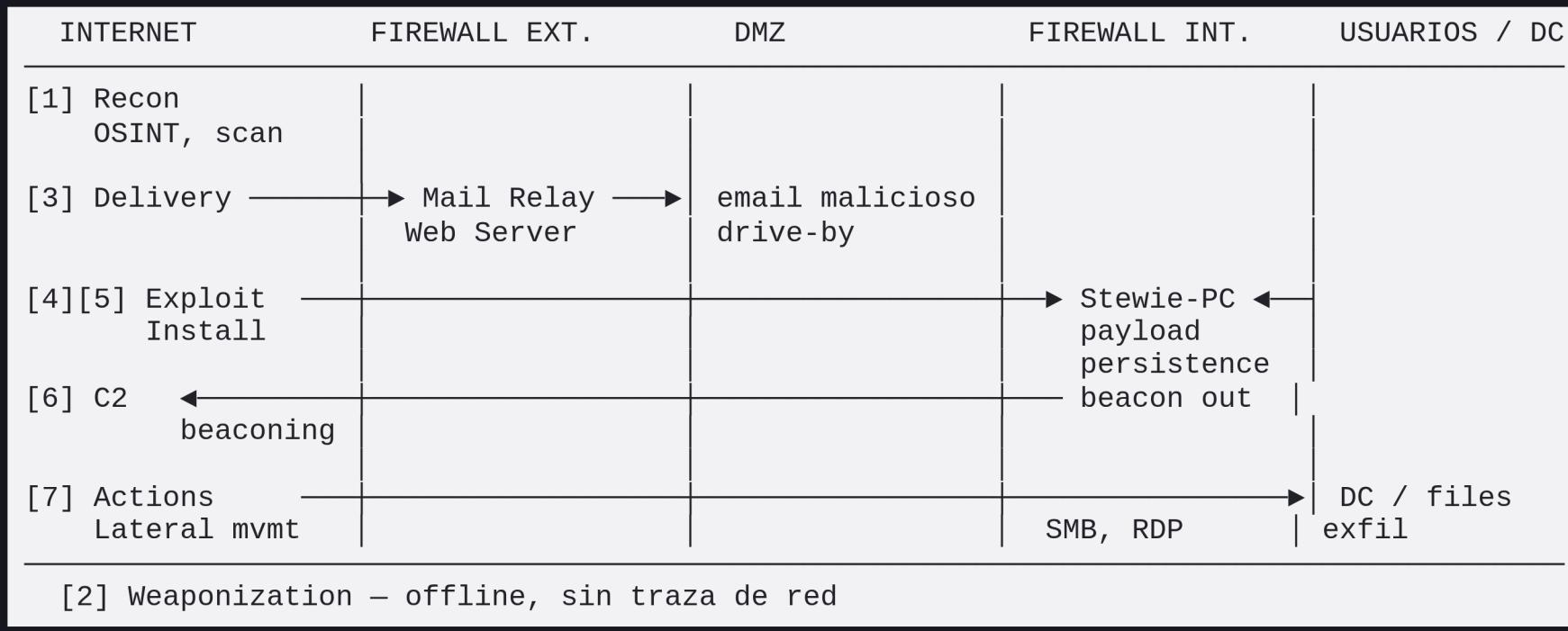
Ana lo envía al servidor AOL, no directamente a 192.168.1.159 — el stream pasa por la infraestructura de AIM

Buscar el paquete con texto legible: "*Here's the secret recipe...*"

# Lab 1 — Buscar la Conversación de IM (LogiCorp)

---

# Kill Chain sobre la Red Corporativa



# Kill Chain → ¿Qué buscamos en el PCAP?

## [1] Reconocimiento

- ▶ Escaneos de puertos desde Internet
- ▶ Peticiones DNS inusuales al dominio

| SYN sin ACK · `nmap` fingerprinting

## [3] Delivery

- ▶ Email con adjunto/URL maliciosa
- ▶ HTTP a dominio recién registrado

| SMTP headers · `http.request` a dominios raros

## [4-5] Exploit & Install

- ▶ Descarga de payload (HTTP/S, DNS)
- ▶ Conexiones a CDNs inusuales

| JA3 fingerprint · PE en stream HTTP

## [6] Command & Control

- ▶ Beaconing periódico (intervalo fijo)
- ▶ DNS tunneling · HTTPS a IP pura

| Entropía de intervalos · `ssl.server_name` ≠ dominio conocido

## [7] Acciones sobre objetivos

- ▶ Lateral movement: SMB, RDP, WMI
- ▶ Exfiltración: volumen anómalo saliente

| Tráfico Este-Oeste inesperado · `smb2` · grandes transfers

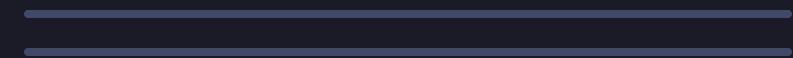
**LogiCorp:** tenemos evidencias de las fases 3, 6 y 7

# Lab 1 — Extraer el documento filtrado (LogiCorp)

---

## Proceso de extracción manual:

- ▶ Localizar la firma del archivo en el dump hexadecimal
- ▶ Archivos ZIP/DOCX comienzan con **PK = 504B0304**
- ▶ Limpiar el HEX: eliminar todo lo anterior y posterior a la firma
- ▶ Guardar como **.docx** → el documento interno de LogiCorp



## El caso Ana — la máquina que recibió el fichero

---

Mientras se investigaba a Ana, el equipo de seguridad analizó el tráfico de la máquina que recibió `recipe.docx`: `annlaptop` ( `192.168.1.159` ).

El PCAP de esa máquina muestra tráfico SMTP saliente en texto claro — una cuenta de correo personal siendo usada desde la red corporativa.

## Lab 2 — LogiCorp: El email de Ann

### Escenario:

El PCAP capturado en el perímetro muestra tráfico **SMTP saliente** desde **annlaptop** (**192.168.1.159**) en texto claro.

Quien lo envía es **Ann Dercover** — la misma máquina que recibió la receta. Está usando su cuenta personal AOL para coordinarse con su contacto externo.

**PCAP:** **Evidencia02.pcap**

### Buscamos:

- ▶ ¿Cuál es el email desde el que escribe Ann?
- ▶ ¿Y su contraseña? (en texto claro en el PCAP)
- ▶ ¿Cuál es el email del contacto externo?
- ▶ ¿Qué dos cosas le pide Ann a su contacto?
- ▶ ¿Cuál es el nombre del fichero adjunto enviado?
- ▶ ¿En qué ciudad y país quedan para encontrarse?

## Lab 2 — Análisis de conversaciones

---

**Estrategia:** Statistics → Conversations → TCP

2 conversaciones — abrimos la primera para analizar si hay correo electrónico

## Lab 2 — Primera conversación

---

Follow TCP Stream → Email a [sec558@gmail.com](mailto:sec558@gmail.com), asunto "*lunch next week*"

Ann dice que no puede quedar. **No es relevante** — es un email de cortesía (o distracción).

## Lab 2 — Segunda conversación (es la correcta)

---

### Follow TCP Stream → AUTH LOGIN:

- ▶ El servidor pide usuario y contraseña en **Base64**
- ▶ c25lYWt5ZzMza0Bhb2wuY29t → sneakyg33k@aol.com
- ▶ NTU4cjAwbHo= → 558r001z
- ▶ Base64 no es cifrado — cualquier decoder online lo lee

### El email a mistersecretx@aol.com:

- ▶ Asunto: *rendezvous*
- ▶ Cuerpo: *"Bring your fake passport and a bathing suit"*
- ▶ Adjunto: secretrendezvous.docx

## Lab 2 — La ciudad del encuentro

---

La ciudad **no está en el cuerpo del email** — está en el documento adjunto

**File** → **Export Objects** → **IMF** → guardamos el **.eml**

- ▶ Abrir el **.eml** y extraer el adjunto
- ▶ El **secretrendezvous.docx** contiene una imagen de Google Maps
- ▶ El mapa muestra la dirección: **Playa del Carmen, México**

# Lab 3 — LogiCorp: Reconocimiento de la red interna

## Escenario:

Se detecta un escaneo activo en el segmento **10.42.42.0/24** de LogiCorp. Uno de los equipos de la red está lanzando peticiones a todos los demás — búsqueda de hosts y puertos abiertos.

Tenemos el PCAP capturado en ese segmento durante la ventana temporal del escaneo.

**PCAP:** Evidencia03.pcap

## Buscamos:

- ▶ ¿Cuál es la IP del equipo desde el que se lanza el escaneo?
- ▶ ¿Qué tipo de escaneo de puertos es el primero?
  - ▶ TCP SYN | TCP ACK | UDP | TCP Connect | TCP XMAS | TCP RST
- ▶ ¿Cuáles son las IPs de los otros sistemas de LogiCorp descubiertos?
- ▶ ¿Cuál es la MAC del equipo Apple encontrado?
- ▶ ¿Cuál es la IP del sistema Windows? \*(pista: TTL de Windows en Google)\*
- ▶ ¿Qué puertos estaban abiertos en el sistema Windows?

## Lab 3 — Identificar el escáner

---

**Statistics** → **Endpoints** — varios equipos involucrados

Por el **número de paquetes enviados**, identificamos el equipo que escanea

Para el tipo de escaneo: filtrar SYN sin ACK de respuesta → **TCP SYN (half-open)**

```
tcp.flags.syn == 1 && tcp.flags.ack == 0
```

## Lab 3 — Tipos de escaneo de puertos

---

Tipo	Flags TCP	Respuesta abierto	Detección
TCP Connect	SYN → SYN/ACK → ACK	Handshake completo	Fácil (logs)
SYN Stealth	SYN → SYN/ACK → RST	Solo SYN/ACK	Moderada
ACK	ACK	RST	Mapeo firewall
XMAS	FIN+PSH+URG	Silencio	Difícil
UDP	UDP vacío	Sin respuesta	Muy difícil
NULL	Sin flags	Silencio	Difícil

# Lab 3 — Resultados

---

## IPs activas y su OS por TTL:

- ▶ Statistics → Endpoints → IPv4
- ▶ TTL = 128 → Windows | TTL = 64 → Linux/Mac
- ▶ El escáner aparece con el recuento de paquetes más alto

## Puertos abiertos en el Windows:

- ▶ Filtro: `tcp.flags == 0x12` (SYN-ACK = puerto abierto)
- ▶ El host Windows responde con SYN-ACK en los puertos que tiene abiertos

## Identificar el equipo Apple:

- ▶ Statistics → Endpoints → Ethernet
- ▶ Los primeros 3 octetos de la MAC = OUI del fabricante
- ▶ `00:16:cb` → Apple Inc. (buscar en *Wireshark OUI lookup*)

Wireshark resuelve los OUI automáticamente en la columna de MACs

# Lab 4 — LogiCorp: Infección de un equipo corporativo

## Escenario:

El análisis del caso LogiCorp nos lleva al origen: ¿cómo entró el atacante?

Un usuario de LogiCorp navegó a un sitio web comprometido. Este PCAP captura la sesión completa: desde la navegación normal hasta la descarga del malware.

**PCAP:** `Evidencia04.pcap` — contraseña del ZIP: `infected`

**Advertencia — hay malware real dentro**

## Buscamos:

- ▶ Fecha y hora exacta de la infección
- ▶ MAC del equipo Windows infectado
- ▶ IP asignada al equipo en el momento de la infección
- ▶ Hostname del equipo
- ▶ ¿Con qué tipo de malware se infectó?

# Lab 4 — Metodología de análisis de malware en PCAP

---

## 1. Identificar el equipo afectado:

- ▶ Statistics → Endpoints → Ethernet → MAC + IP del cliente
- ▶ Hostname: filtro `dhcp.option.hostname` → aparece en el campo Option 12
- ▶ O añadir columna *Host* en Wireshark para verlo inline

## 2. Identificar el momento de la infección:

- ▶ Buscar la primera descarga sospechosa
- ▶ HTTP GET de `.exe`, `.dll`, `.zip`, `.js`, `.ps1`
- ▶ Respuesta `Content-Type: application/x-msdownload` = ejecutable

## 3. Identificar el malware:

- ▶ Exportar el objeto HTTP descargado
- ▶ Calcular el hash MD5/SHA256
- ▶ Consultar VirusTotal, MalwareBazaar
- ▶ Analizar comportamiento de red post-infección

Si aparece dominio con caracteres aleatorios  
= **DGA** (Domain Generation Algorithm)

# LogiCorp — Infección confirmada

## Aplicando la metodología anterior al caso LogiCorp:

### 1. Equipo afectado:

- ▶ Statistics → Endpoints → Ethernet
- ▶ Hostname en DHCP: Stewie-PC
- ▶ IP en el momento de la infección:  
172.16.4.193

### 2. Momento de infección:

- ▶ Visita a www.homeimprovement.com (sitio comprometido)
- ▶ Redirige a exploit kit → descarga application/x-msdownload
- ▶ Primer EXE descargado de 194.87.234.129 a las 22:55 UTC

### 3. Malware identificado:

- ▶ C2 post-infección: spotsbill.com
- ▶ Página de pago Bitcoin:  
p27dokhpz2n7nvgr.1jw2lx.top
- ▶ Resultado: **Ransomware** (patrón Cerber)

**Timeline:** homeimprovement.com → exploit kit  
→ EXE desde 194.87.234.129 → ransomware  
→ C2 spotsbill.com

# Flujos de Información (Network Flows)

---

**NetFlow / IPFIX / sFlow**

**Telemetría de red a escala**

# Análisis de Flujos — Definición

---

## ¿Qué es un flow (flujo)?

Resumen de tráfico **unidireccional** que comparte:

- ▶ IP origen
- ▶ IP destino
- ▶ Puerto origen
- ▶ Puerto destino
- ▶ Protocolo

*RFC 3954: "secuencia unidireccional de paquetes con alguna propiedad común"*

## Los datos de un flujo contienen:

- ▶ Los 5 elementos de la quíntupla
- ▶ Flags TCP de la sesión
- ▶ Bytes y paquetes totales transferidos
- ▶ Hora de inicio, fin y duración
- ▶ Sensor que recolectó el flujo

Un flujo ≠ una conexión TCP — puede haber múltiples conexiones en un flujo o viceversa

# Análisis de Flujos — Usos

---

## El análisis de flujos se usa para:

- ▶ Identificar patrones en el tráfico
- ▶ Aislar actividad sospechosa
- ▶ Analizar protocolos de capas superiores
- ▶ Extraer información sobre comunicaciones

## Estándares:

- ▶ **NetFlow v5/v7/v9** — Cisco (1996), ahora IETF
- ▶ **IPFIX** — "NetFlow v10", estándar abierto
- ▶ **sFlow** — muestreo con menor granularidad
- ▶ **jFlow** — implementación Juniper de sFlow

# NetFlow v9 — Formato

---

# PCAP vs \*Flows — La Gran Diferencia

---

## PCAP

- ▶ Como **escuchar en un hilo telefónico**
- ▶ Todo el contenido — cabeceras + payload
- ▶ Muy costoso en almacenamiento
- ▶ Retención limitada (días/semanas)

## \*Flows

- ▶ Como **mirar la factura del teléfono**
- ▶ Solo metadatos — quién habló con quién
- ▶ Muy eficiente en almacenamiento
- ▶ Retención larga (meses/años)

Se puede aprender mucho de la "factura":

Quién → quién | protocolo | duración | velocidad | dirección de la transferencia

# SNMP vs \*Flow

---

## SNMP (polling)

- ▶ El gestor solicita información al dispositivo
- ▶ Necesita decidir **cuándo** hacer el poll
- ▶ Para cuando se hace el poll, la info puede no estar
- ▶ La correlación requiere múltiples peticiones
- ▶ El equipo no controla la cantidad de información

## NetFlow (push)

- ▶ La información se manda **de forma asíncrona**
- ▶ Postprocesado posible en el router/switch
- ▶ La información se borra del equipo tras exportar
- ▶ Escalable — cada router/switch es un sensor

Los flows son la forma de **telemetría** enviada por routers y switches — cada uno es un sensor de red

# De Dónde se Obtienen los Flows

## Fuentes:

- ▶ Del **router o switch** directamente (NetFlow/sFlow)
- ▶ Generados a partir de un **PCAP existente**
- ▶ Generados por **probes de red** que analizan el tráfico en tiempo real

## Limitaciones:

- ▶ Capacidad del enlace analizado
- ▶ Recursos de hardware del dispositivo
- ▶ Muestreo (sFlow) → no todos los paquetes
- ▶ Similares a los desafíos del despliegue de Arkime

## Filtrado

- ▶ Básico para reducir el espacio de análisis
- ▶ Aislar actividad por IP específica
- ▶ Filtrar por patrones de tráfico conocidos
- ▶ Usar un pequeño porcentaje para análisis detallado

## Baseline

- ▶ Los flows permiten mayor retención de datos
- ▶ Construir perfil de actividad "normal"
- ▶ Detectar cambios drásticos en el comportamiento de un host

## Valores no deseados

- ▶ Similar a las reglas de un IDS
- ▶ Lista de IPs, puertos o protocolos sospechosos

## Búsqueda de patrones

- ▶ Segundo IP, puertos, protocolos
- ▶ Intentos de conexión y escaneo de puertos
- ▶ Transferencias grandes y dirección de la transferencia

Anomalía clásica: Un host que normalmente envía 1MB/día de repente envía 10GB hacia el exterior

# Full Packet Capture (FPC)

---

**La captura total de paquetes**

**Máxima visibilidad, máximo coste**

# Necesidad del FPC

---

## ¿Por qué FPC?

Casi todas las herramientas de seguridad usan un modelo de **seguridad negativa**:

- ▶ Difícilmente detectan Zero-Days
- ▶ Fallan con malware nuevo (sin firmas)
- ▶ No detectan nada si no tienen una firma previa

## Ventajas del FPC:

- ▶ Permite revisar **todas las comunicaciones** de todos los sistemas
- ▶ Detecta comunicaciones maliciosas que evaden otras herramientas
- ▶ **Retrospección** — reproducir tráfico antiguo con nuevas firmas
- ▶ Determinar si hubo ataque **antes del parche**
- ▶ Extraer nuevas firmas y muestras de malware del tráfico

# Planificación del FPC

---

## Preguntas clave al desplegar:

- ▶ ¿Dónde lo colocamos en la red?
- ▶ ¿Qué va a monitorizar?
- ▶ ¿Cuáles son las necesidades de **retención**?
- ▶ ¿Qué hay de la **redundancia y escalabilidad**?
- ▶ ¿Herramienta **comercial o open source**?

## Colocación recomendada:

- ▶ En los **límites entre redes confiables y no confiables**
  - ▶ Entre LAN e Internet
  - ▶ Entre Internet y la DMZ
  - ▶ Entre segmentos críticos internos

FPC detrás de un FW que actúa como proxy  
→ solo ve tráfico del proxy, no del cliente real

# Necesidades de Almacenamiento FPC

## Factores a considerar:

- ▶ ¿Existe obligación legal de retención?
- ▶ ¿Cuál es el **tiempo medio de detección** (MTTD)?
- ▶ ¿SOC 24/7 o solo en horario de oficina?
- ▶ MTTD determina el **\*\*mínimo\*\*** tiempo de almacenamiento

## Cálculo básico:

Capacidad del enlace  
  × ocupación media del enlace  
  × segundos de almacenamiento  
= bytes necesarios

## Ejemplo:

$$0.75 \text{ Gbps} \times 75\% \text{ uso} \times 72\text{h retención}$$

$$= 0.75 \times 0.75 \times 72 \times 3600 / 8 / 1024^3$$

$$\approx \mathbf{\sim 24 \text{ TB}} \text{ de almacenamiento}$$

**Ojo:** Además del espacio hay que tener en cuenta la **velocidad de escritura** de los discos

→ Puede requerir combinación SSD + HDD o RAID

## Network TAPs (preferido)

- ▶ Método preferido para FPC
- ▶ TAP físico — tráfico idéntico al original
- ▶ No requiere SPAN port
- ▶ No descarta paquetes en caso de saturación

## NICs de captura

- ▶ Tarjetas con chips especializados
- ▶ Capaces de procesar tráfico de alta velocidad
- ▶ Con timestamping hardware de alta precisión

## Filtrado BPF en captura

- ▶ No todo el tráfico en el TAP es útil
  - ▷ Tráfico del propio FPC
  - ▷ Tráfico de backup
- ▶ Filtros BPF en las NICs de captura
- ▶ Soportados por OpenFPC, Snort, Suricata, Arkime

## Packet Brokers

- ▶ Para tráfico excesivamente alto
- ▶ Balanceo de carga entre sensores
- ▶ Filtrado L2-L7 y descifrado

# Arkime (antes Moloch)

## Sistema open-source para FPC masivo:

- ▶ Captura, indexa y permite búsqueda en PCAPs
- ▶ Escala a terabytes de tráfico histórico
- ▶ Búsqueda en segundos gracias a Elasticsearch

## Tres componentes:

- ▶ **Capture** — almacena PCAPs completos en disco
- ▶ **Elasticsearch** — indexa metadatos de sesiones (SPI)
- ▶ **Viewer** — interfaz web para búsqueda y análisis

```
# Búsquedas en Arkime (ejemplos):  
  
# TLS 1.3 a IPs rusas  
protocols == tls &&  
  tls.version == "TLSv1.3" &&  
  country == "RU"  
  
# DNS over HTTPS (DoH)  
host == cloudflare-dns.com ||  
  host == dns.google  
  
# Beaconing detectado  
packets >= 50 &&  
  bytes < 10000 &&  
  duration > 3600  
  
# Exfiltración ICMP  
protocols == icmp &&  
  bytes.dst > 100000
```

# Arkime — Información SPI

---

**Session Profile Information — metadatos extraídos de cada sesión:**

## DNS

- ▶ Direcciones IP resueltas
- ▶ Hostnames consultados

## HTTP

- ▶ Método (GET/POST/PUT...)
- ▶ Códigos de estado
- ▶ Cabeceras (User-Agent, Host, Referer)
- ▶ Tipo de contenido

## TLS/SSL

- ▶ Certificados (sujeto, emisor, SANs)
- ▶ Números de serie
- ▶ JA3/JA4 fingerprints

## SSH

- ▶ Nombre del cliente y versión
- ▶ Clave pública del servidor

## SMTP

- ▶ Cabeceras de correo (From, To, Subject)
- ▶ Asunto y tipo de contenido

# Integración con el Stack de Seguridad Moderno

---

## Pipeline de Detección y Respuesta

### 1. Captura

- ▷ TAP/SPAN → Arkime (PCAP indexado)
- ▷ Zeek/Suricata (logs enriquecidos)

### 2. Agregación

- ▷ Flows → Elastic Stack / Splunk / Chronicle

### 3. Detección

- ▷ SIEM Rules + ML → Alertas de anomalías
- ▷ Beaconing, DGA, lateral movement

### 4. Investigación

- ▷ Alerta SIEM → Pivot a Arkime con timestamp
- ▷ Extraer PCAP de contexto

### 5. Enriquecimiento

- ▷ IOCs → VirusTotal, AbuseIPDB, ThreatFox
- ▷ Correlación con EDR (proceso, usuario)

### 6. Respuesta

- ▷ Firewall API → Bloqueo automático
- ▷ SOAR → Ticket de incidente + notificación

**Clave:** PCAP es el 'ground truth' cuando el SIEM duda

## Expresiones regulares para detección de amenazas

# Regex Cheat Sheet para Wireshark

```
# Archivos ejecutables descargados  
http.request.uri matches  
"\.(exe|dll|ps1|bat|vbs)$"  
  
# DGA - Dominios generados algorítmicamente  
dns.qry.name matches  
"^[a-z]{15,}\.(com|net|org)$"  
  
# Credenciales en texto claro  
frame matches  
"(?i)(password|passwd|pwd)=.{3,}"
```

```
# Tráfico de bots automatizados  
http.user_agent matches  
"(?i)(bot|crawler|spider|scrapy)"  
  
# SQL Injection en URLs  
http.request.uri matches  
"(?i)(union|select|from|where)"  
  
# Exfiltración Base64 vía DNS  
dns.qry.name matches  
"^[A-Za-z0-9+/]{30,}="
```

(?i) = Case Insensitive — úsalo **siempre** en user-agents

# JA3/JA4 — TLS Client Fingerprinting

## ¿Qué es JA3/JA4?

- ▶ Huella digital del **TLS ClientHello**
- ▶ Basada en: versión, cipher suites, extensiones, curvas elípticas
- ▶ Identifica la implementación TLS del cliente
- ▶ Funciona **sin descifrar el tráfico**

## Malware conocido:

- ▶ Cobalt Strike:  
e7d705a3286e19ea42f587b344ee6865
- ▶ Sliver C2: 51c64c77e60f3980eea90869b68c58a8
- ▶ Trickbot: 6734f37431670b3ab4292b8f60f29984

## Herramientas:

- ▶ ja3er.com — base de datos de hashes JA3
- ▶ Zeek con ja3.zeek script
- ▶ Arkime incluye JA3 en el SPI
- ▶ Wireshark muestra el ClientHello en detalle

JA4 es la evolución de JA3 — más resistente a trivialidades como reordenar cipher suites

## Domain Fronting 2.0

- ▶ SNI dice `cloudflare.com`
- ▶ HTTP Host header apunta al C2
- ▶ El tráfico parece ir a Cloudflare
- ▶ Detección: Comparar SNI vs Host

## DNS over HTTPS (DoH)

- ▶ Resoluciones C2 vía `dns.google`
- ▶ Parece tráfico HTTPS legítimo
- ▶ Imposible bloquear sin rompimiento TLS

## Protocol Tunneling

- ▶ SSH sobre HTTP
- ▶ VPN sobre ICMP
- ▶ C2 sobre DNS (DNS tunneling)

## Time-based Evasion

- ▶ Beaconing con **jitter aleatorio**
- ▶ Evita la detección por intervalos fijos
- ▶ Solo el análisis estadístico lo detecta

Todas estas técnicas abusan de protocolos legítimos — bloquear el protocolo bloquearía tráfico normal

# APT Kill Chain — Indicadores de Red

Fase	Indicadores en PCAP
Reconnaissance	DNS masivas, patrones Shodan, SYN a rangos de IP
Delivery	HTTP download .doc/.zip, User-Agent de herramienta
Exploitation	Payload anómalo, respuestas de error del servidor
C2	HTTPS con JA3 malicioso, DGA domains, beaconing
Lateral Movement	SMB admin\$, RDP, WMI entre hosts internos
Exfiltration	DNS tunneling, uploads grandes, ICMP con payload

## Regla práctica:

Cada fase deja huellas distintas — el PCAP es el registro completo de la actividad del atacante

APTs sofisticados operan durante **meses** antes de la detección — la retención de flows históricos es crítica

# Detecting Lateral Movement

## Protocolos de movimiento lateral:

- ▶ **SMB (445/tcp)**
  - ▷ Acceso a shares admin\$, C\$, IPC\$
  - ▷ Transferencia de herramientas
- ▶ **RDP (3389/tcp)**
  - ▷ Login desde servidor interno → sospechoso
- ▶ **WMI (135/tcp)**
  - ▷ Remote command execution
- ▶ **WinRM (5985/tcp)**
  - ▷ PowerShell remoting

### Red Flag clásico:

Servidor web → RDP → Domain Controller

Un servidor web **nunca** debería iniciar una sesión RDP hacia el DC

### Detección en Wireshark:

```
tcp.port == 445 && smb2
```

```
tcp.port == 3389 && ip.src != [known admin  
IPs]
```

# Encrypted Traffic Analysis (ETA)

---

## Detectar malware SIN descifrar payload

### Técnicas de Machine Learning:

#### ▶ Packet Size Distribution

- ▷ Malware C2: distribución uniforme (mensajes fijos)

#### ▶ Inter-Arrival Times

- ▷ C2 beaconing: intervalos constantes o con jitter mínimo

#### ▶ TLS Certificate Analysis

- ▷ Self-signed, CN mismatch, expirado

#### ▶ Flow Duration vs Bytes

- ▷ Flujo muy largo con muy pocos datos → beaconing

### Herramientas:

- ▶ **Joy** (Cisco) — análisis estadístico de flows cifrados
- ▶ **Mercury** — fingerprinting de protocolos cifrados
- ▶ **Zeek ML** — scripts de detección basados en ML
- ▶ **RITA** — detección de beaconing en logs Zeek

ETA no reemplaza el descifrado, pero permite priorizar qué descifrar

# QUIC / HTTP/3 — El Nuevo Reto

## ¿Qué es HTTP/3?

- ▶ HTTP/3 corre sobre **QUIC** (UDP)
- ▶ El 50%+ del tráfico web en 2025
- ▶ Usado por: Google, Facebook, Cloudflare

## Desafíos forenses:

- ▶ No hay TCP handshake (sin SYN/ACK)
- ▶ Connection ID ofuscado
- ▶ 0-RTT: la primera request ya va cifrada
- ▶ Wireshark puede diseccionarlo pero necesita las keys

**Los filtros TCP clásicos no funcionan con QUIC:**

```
tcp.port == 443 no captura QUIC  
udp.port == 443 sí
```

## Estrategia:

- ▶ SSLKEYLOGFILE en el endpoint
- ▶ Análisis estadístico del flujo UDP
- ▶ Correlación con logs DNS

# Memory Forensics + PCAP Correlation

## El poder de la correlación:

Cuando tenemos volcado de memoria **Y** PCAP, podemos:

- ▶ Identificar el **proceso exacto** que generó el tráfico malicioso
- ▶ Reconstruir la cadena completa: proceso → conexión → payload
- ▶ Confirmar si el proceso fue inyectado

## Pipeline:

- ▶ Volcado de memoria → Volatility3:  
`windows.netscan`
- ▶ Extraer: PID, proceso, IP:puerto
- ▶ Correlacionar con PCAP por IP:puerto y timestamp
- ▶ Identificar el proceso malicioso
- ▶ Reconstruir el timeline completo

## Ejemplo real:

Memory: `powershell.exe (PID:4832) → 185.220.101.45:443`

PCAP confirma: HTTP POST a  
`185.220.101.45:443` con User-Agent de  
PowerShell

→ Conclusión: PowerShell fue el dropper

Sin correlación entre fuentes, un atacante con OPSEC puede confundir la atribución

# LogiCorp — Búsqueda histórica con Arkime

## Ejemplo de búsqueda histórica:

Con un IOC identificado — por ejemplo `spotsbill.com` del Lab 4 — buscamos en Arkime **cuándo empezó realmente** la actividad del malware.

## Query en Arkime:

```
host == "spotsbill.com"
```

## Resultado hipotético:

- ▶ Primera sesión: días o semanas antes del reporte del usuario
- ▶ El malware llevaba tiempo activo sin ser detectado
- ▶ Permite reconstruir el timeline completo de la infección

## Si tuviéramos EDR desplegado:

- ▶ Arkime identifica IP y puerto → cruzamos con logs de EDR
- ▶ El EDR revelaría el proceso responsable de la conexión
- ▶ Confirmaría el mecanismo de ejecución (process injection, service, etc.)

## El valor de combinar fuentes:

PCAP + Arkime + EDR = atribución completa y timeline de 23 días que el SIEM solo no habría detectado

# Caso LogiCorp — Resolución

## ¿Qué encontramos en los PCAPs?

- ▶ Lab 1: `recipe.docx` transferida por AIM desde `Sec558user1` (Ana)
- ▶ Lab 2: Ann Dercover usa `sneakyg33k@aol.com` para coordinar cita en Playa del Carmen con `mistersecretx@aol.com`
- ▶ Lab 3: Escaneo TCP SYN desde `10.42.42.253` — Windows en `.50` (puertos 135, 139), Apple en `.25`
- ▶ Lab 4: Stewie-PC infectado vía `www.homeimprovement.com` → exploit kit → **ransomware**

## Herramientas que usamos:

- ▶ Wireshark — Follow TCP Stream, Export Objects
- ▶ Statistics → Endpoints, Conversations
- ▶ VirusTotal / MalwareBazaar para identificación

## Conclusión forense:

Insider threat (Ana + Ann) filtró la receta secreta y coordinó un encuentro externo. Paralelamente, un equipo corporativo ( Stewie-PC , 172.16.4.193 ) fue infectado con ransomware tras visitar un sitio comprometido.

## IOCs del Lab 4:

- Servidor malware: `194.87.234.129`
- C2 post-infección: `spotsbill.com`
- Ransom page:  
`p27dokhpz2n7nvgr.1jw2lx.top`

SHA-256 de cada PCAP documentado.  
Evidencia preservada para el equipo legal.

# Próximos Pasos en tu Carrera

## Si estás empezando

- ▶ **Malware-Traffic-Analysis.net** — PCAPs de malware reales, actualizados semanalmente
- ▶ **Wireshark** — práctica con capturas propias de tu red
- ▶ **CTFs** — TryHackMe, HackTheBox, PicoCTF

## Si ya te interesa en serio

- ▶ **PacketTotal** — análisis colaborativo de PCAPs
- ▶ **SANS ISC** — lectura diaria de amenazas actuales
- ▶ **BSides** — eventos locales de seguridad
- ▶ Escribe tus análisis — blog propio o GitHub write-ups

## Si quieres dedicarte profesionalmente

- ▶ **SANS FOR572** → examen **GCIA**
  - ▶ El binomio estándar del sector — el curso prepara el examen
- ▶ Busca roles: "*network analyst*", "*tier 2 SOC*", "*DFIR junior*"

El camino más corto: un PCAP real analizado y documentado en público vale más en una entrevista que cualquier curso sin práctica.

# Resumen Final

## Lo que hemos aprendido:

- ▶ Fundamentos de protocolos de red (L2-L7)
- ▶ Captura y análisis con Wireshark / tcpdump
- ▶ Filtros BPF y display filters avanzados
- ▶ Metodología de forense de red
- ▶ Fuentes de información en la investigación
- ▶ NetFlow y análisis de flujos
- ▶ Full Packet Capture con Arkime
- ▶ Técnicas avanzadas: JA3, ETA, QUIC

**El mejor analista de forense de red es el que:**

- ▶ Conoce bien los protocolos
- ▶ Sabe qué es normal en su red
- ▶ Practica con PCAPs reales
- ▶ Correlaciona múltiples fuentes
- ▶ Documenta todo con rigor

**Contacto:** mhercol[@]gmail[.]com

## Network Forensics 2025

**Gracias — preguntas y contacto:**

mhercol[@]gmail[.]com

**A continuación:** Material de referencia y módulo avanzado (bonus)

Glosario · Apéndices · Cloud & Kubernetes · eBPF

## BONUS TRACK – Infraestructuras Modernas

Este módulo va más allá del curso base.  
Es material de referencia para cuando trabajes  
en entornos cloud o con contenedores.

No es examen. Es contexto profesional real.

## **Cloud, contenedores y microservicios: visibilidad sin acceso físico**

La infraestructura moderna ha reescrito las reglas del forense de red

## AWS

- ▶ **VPC Flow Logs**: Metadatos (5-tupla, bytes, packets)
- ▶ **VPC Traffic Mirroring**: PCAP completo -> EC2/NLB

## Azure

- ▶ **NSG Flow Logs**: Equivalente a VPC Flow Logs
- ▶ **Network Watcher Packet Capture**: PCAP temporal (máx 5h, 10GB)

## GCP

- ▶ **VPC Flow Logs**: Muestreo configurable (1:1 a 1:1000)
- ▶ **Packet Mirroring**: Clona tráfico a collector instance

### [!] Limitación crítica

Flow Logs **NO contienen payload**  
(solo headers L3/L4)

# El nuevo paradigma: contenedores vs VMs

**"Trata a los contenedores como ganado, no como mascotas"**

Los servidores tradicionales son *mascotas*: tienen nombre, historia y se reparan cuando fallan.  
Los contenedores son *ganado*: fungibles, numerados y reemplazados — nunca reparados. Esta filosofía tiene consecuencias directas en el forense: **la evidencia es efímera por diseño.**

## Infraestructura tradicional (VMs)

- ▶ IP fija, MAC conocida, OS persistente
- ▶ Logs en disco, en el mismo nodo
- ▶ Ciclo de vida: semanas/meses
- ▶ Estado forense disponible post-incidente

## Herramientas clásicas funcionan:

`[tcpdump]`, Wireshark, `[netstat]`, PCAP directo

## Entornos contenerizados

- ▶ IP efímera — cambia en cada reinicio
- ▶ Contenedor destruido = evidencia destruida
- ▶ Ciclo de vida: segundos/minutos
- ▶ Un pod puede vivir en cualquier nodo

## Los retos forenses son nuevos:

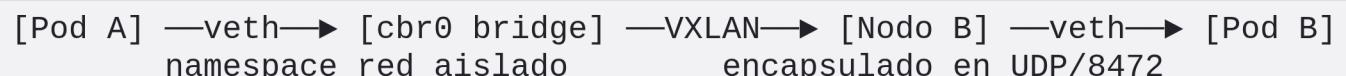
visibilidad de red, namespaces, overlay, mTLS

# Arquitectura de red en Kubernetes

**Problema central:** en Kubernetes, el tráfico entre pods viaja por redes virtuales superpuestas (*overlay networks*) — invisible a herramientas de captura convencionales sobre la NIC física

## Capas de red involucradas:

- ▶ \*\*eth0 del pod\*\* — interfaz virtual dentro del namespace de red del contenedor
- ▶ \*\*veth pair\*\* — cable virtual entre el pod y el nodo host
- ▶ \*\*bridge/CNI\*\* — plugin de red: Flannel, Calico, Cilium, Weave
- ▶ \*\*NIC física del nodo\*\* — tráfico encapsulado (VXLAN, Geneve, IP-in-IP)
- ▶ \*\*kube-proxy / eBPF\*\* — balanceo de servicios (iptables o eBPF según CNI)



# Desafíos de visibilidad — Por qué el PCAP clásico falla

---

## Problema 1: Namespaces de red

Cada pod tiene su propio namespace de red. `[tcpdump]` en el host **no ve** el tráfico interno entre pods del mismo nodo.

## Problema 2: Ephemeral by design

Un `kubectl delete pod` destruye toda evidencia local. Los contenedores son inmutables — no se "parchean", se recrean.

## Problema 3: IPs dinámicas

DHCP/IPAM asigna IPs al vuelo. La IP `[10.0.1.42]` que ves en logs puede pertenecer a 10 pods distintos en el mismo día.

## Problema 4: Encapsulación

El tráfico Este-Oeste entre nodos viaja encapsulado en VXLAN (UDP/8472) o Geneve. Capturar en la NIC física muestra **encapsulación**, no el payload real.

## Problema 5: Escala y velocidad

Miles de pods, decenas de nodos. Retención muy corta. Correlacionar un evento pasado con logs de pods ya destruidos es extremadamente difícil.

## Problema 6: mTLS en service meshes

Istio/Linkerd cifran **todo** el tráfico Este-Oeste con mTLS. Sin gestión de claves, el tráfico es opaco.

# Redes overlay — El tráfico "invisible"

**VXLAN (Virtual eXtensible LAN)**: el tráfico entre pods de distintos nodos se encapsula en UDP/8472. Un `tcpdump` en la NIC física del nodo captura el wrapper, no el contenido real.

```
# Lo que ves capturando en eth0 del nodo:  
IP nodo-1:47392 > nodo-2:8472: VXLAN, flags [I], vni 1  
IP 10.0.1.5:54321 > 10.0.2.8:443: Flags [P.], ...  
#      ^pod-origen          ^pod-destino    ^payload dentro del VXLAN  
  
# Para ver el tráfico real, necesitas capturar DENTRO del namespace del pod:  
nsenter -t <PID_del_pod> -n -- tcpdump -i eth0 -w /tmp/captura.pcap
```

## CNI plugins y sus implicaciones forenses:

- ▶ **Flannel** — VXLAN simple, fácil de descifrar manualmente
- ▶ **Calico** — IP-in-IP o BGP nativo, más eficiente, misma opacidad
- ▶ **Cilium** — eBPF nativo, ofrece su propio plano de observabilidad (Hubble)
- ▶ **Weave** — cifrado opcional del overlay (añade otra capa de opacidad)

# Service Mesh y mTLS — El reto del cifrado Este-Oeste

---

## Istio / Linkerd en modo mTLS

Cada pod tiene un sidecar proxy (**Envoy**) que intercepta todo el tráfico entrante y saliente y aplica mTLS automáticamente.

### Consecuencia forense:

El tráfico entre `servicio-A` → `servicio-B` está cifrado aunque ambos servicios estén en la misma red interna.

### A diferencia de TLS externo:

No hay un único punto de terminación TLS (como un proxy inverso). Las claves están distribuidas entre todos los sidecars.

## Estrategias de visibilidad:

- ▶ **\*\*Envoy access logs\*\*** — el sidecar registra conexiones (IP, puerto, L7 si hay permiso)
- ▶ **\*\*Istio telemetry API\*\*** — exporta métricas y trazas a Prometheus/Jaeger
- ▶ **\*\*Captura en el sidecar\*\*** — `kubectl exec` en el contenedor `istio-proxy` para tcpdump antes del cifrado
- ▶ **\*\*Mutual TLS certs\*\*** — los certificados de corta vida (SVID con SPIFFE) son trazables por identidad de workload, no por IP

```
# Captura en el sidecar Envoy (tráfico pre-mTLS):  
kubectl exec -it <pod> -c istio-proxy -- \  
tcpdump -i lo -w /tmp/cap.pcap
```

# Captura de tráfico en Kubernetes — nsenter

**nsenter**: entra en los namespaces de Linux (red, PID, mount) de un proceso en ejecución. Permite ejecutar herramientas del host **dentro del namespace de red de un pod** sin modificar el pod.

```
# 1. Obtener el PID del proceso principal del contenedor en el nodo
CONTAINER_ID=$(kubectl get pod <pod-name> -o jsonpath='{.status.containerStatuses[0].containerID}' \
    | cut -d'/' -f3)
PID=$(docker inspect --format '{{.State.Pid}}' $CONTAINER_ID)
# o con containerd:
PID=$(crictl inspect $CONTAINER_ID | jq '.info.pid')

# 2. Entrar en su namespace de red y capturar
nsenter -t $PID -n -- tcpdump -i eth0 -w /tmp/pod-capture.pcap

# 3. Copiar la captura al host
# (el archivo ya está en /tmp del sistema de archivos del host, no del pod)
```

**Ventajas:** no requiere modificar el pod, sin privilegios extra en el pod, funciona con contenedores sin shell

**Limitación:** el pod debe estar **vivo** en el momento de la captura

# ksniff — Captura de tráfico remota desde kubectl

**ksniff** es un plugin de `kubectl` que automatiza la captura de tráfico en un pod remoto y lo redirige en tiempo real a Wireshark local — sin necesidad de SSH ni privilegios en el pod.

```
# Instalación
kubectl krew install sniff

# Captura en tiempo real (abre Wireshark localmente):
kubectl sniff <pod-name> -n <namespace>

# Captura a fichero PCAP:
kubectl sniff <pod-name> -n <namespace> -o /tmp/pod-traffic.pcap

# Filtro BPF para reducir ruido:
kubectl sniff <pod-name> -n <namespace> -f "port 8080"

# Pod sin privilegios (usa contenedor efímero):
kubectl sniff <pod-name> -n <namespace> --privileged=false
```

## Casos de uso forenses:

- ▶ Capturar tráfico de un pod sospechoso sin interrumpir el servicio
- ▶ Validar si un pod está realizando conexiones no autorizadas
- ▶ Analizar tráfico de microservicio en un entorno de producción

# eBPF — La revolución de la visibilidad en contenedores

## ¿Qué es eBPF?

Extended Berkeley Packet Filter permite ejecutar programas sandboxeados **en el kernel de Linux** sin modificar el código del kernel.

## Para forense de contenedores:

- ▶ Captura de tráfico \*\*antes\*\* del cifrado TLS
- ▶ Trazado de syscalls por PID/contenedor
- ▶ Visibilidad de conexiones de red a nivel de proceso
- ▶ Overhead mínimo — apto para producción
- ▶ No requiere modificar pods ni sidecars

## Herramientas basadas en eBPF:

Herramienta	Uso forense
<b>Falco</b>	Detección de anomalías en tiempo real
<b>Tetragon</b>	Trazado de procesos y red (Cilium)
<b>Hubble</b>	Observabilidad de red en Kubernetes
<b>Pixie</b>	Análisis de tráfico L7 sin instrumentación
<b>Tracee</b>	Forense de syscalls (Aqua Security)
<b>bpftrace</b>	Scripting eBPF ad-hoc

```
# Ver conexiones TCP de todos los pods en tiempo real:  
bpftrace -e 'kprobe:tcp_connect {  
    printf("%s -> %s\n", comm,  
        ntop(arg0->__sk_common.skc_daddr)); }'
```

# Falco — Detección en tiempo real en contenedores

**Falco** (CNCF) usa eBPF/kernel module para monitorizar llamadas al sistema en todos los contenedores del nodo. Define reglas declarativas para detectar comportamientos anómalos.

## Reglas relevantes para forense de red:

```
# Detección: shell dentro de contenedor de producción
- rule: Terminal shell in container
  desc: Container running an interactive shell
  condition: spawned_process and container and shell_procs and proc.tty != 0
  output: "Shell en contenedor (user=%user.name container=%container.name)"
  priority: WARNING

# Detección: conexión saliente inesperada desde contenedor
- rule: Unexpected outbound connection
  desc: Outbound connection on unexpected port
  condition: outbound and container and not (fd.sport in (allowed_ports))
  output: "Conexión no autorizada (pod=%k8s.pod.name dst=%fd.rip:%fd.rport)"
  priority: CRITICAL
```

## Exportación de alertas:

- ▶ stdout → log aggregation (Fluentd, Loki)
- ▶ gRPC → Falcosidekick → Slack, SIEM, PagerDuty
- ▶ JSON estructurado con metadatos Kubernetes (pod, namespace, node, image)

# Cilium Hubble — Observabilidad de red en Kubernetes

**Hubble** es el plano de observabilidad de Cilium. Usando eBPF proporciona visibilidad L3/L4/L7 del tráfico entre pods **sin modificar aplicaciones ni añadir sidecars**.

## Capacidades:

- ▶ Flujos de red por pod, namespace, label
- ▶ Visibilidad L7: HTTP, gRPC, DNS, Kafka
- ▶ Mapa de servicios en tiempo real
- ▶ Correlación con identidades Kubernetes
- ▶ Retención configurable de flujos

```
# Instalar CLI de Hubble
hubble version

# Ver flujos en tiempo real (todos los namespaces):
hubble observe --all-namespaces

# Filtrar por pod concreto:
hubble observe --pod frontend/nginx-7d9f --follow

# Ver solo conexiones rechazadas (policy violations):
hubble observe --verdict DROPPED

# Flujos DNS para detectar C2:
hubble observe -t 17 --protocol DNS \
  --namespace production

# Exportar a JSON para análisis forense:
hubble observe --output json > flujos-$(date +%Y%m%d).json
```

# Forense de sistema de archivos — OverlayFS

Los contenedores Docker/containerd usan **OverlayFS**: capas de imagen de solo lectura + capa de escritura efímera (upperdir). Al destruir el contenedor, la capa de escritura desaparece.

## Estructura OverlayFS:

```
/var/lib/docker/overlay2/  
  <layer-id>/  
    diff/      ← cambios del contenedor (upperdir)  
    merged/   ← vista combinada (mountpoint)  
    work/     ← directorio interno OverlayFS  
    lower     ← referencia a capas inferiores
```

## Evidencia recuperable (si el contenedor aún existe):

- ▶ Archivos descargados por malware
- ▶ Credenciales escritas en disco
- ▶ Scripts ejecutados y logs internos

```
# Extraer sistema de archivos de un contenedor vivo:  
docker export <container-id> > container-fs.tar  
  
# Inspeccionar la capa de escritura directamente:  
LAYER=$(docker inspect <id> \  
  --format '{{.GraphDriver.Data.UpperDir}}')  
ls -la $LAYER  
  
# Copiar archivo específico del contenedor:  
docker cp <container-id>:/tmp/malware.sh ./evidencia/  
  
# Si ya está destruido – buscar en capas de imagen:  
docker history <image>:tag  
docker save <image> | tar -xv
```

## Cadena de custodia:

```
# Hash antes de cualquier análisis:  
sha256sum container-fs.tar > container-fs.tar.sha256
```

# Kubernetes Audit Logs — La traza del plano de control

El **API Server de Kubernetes** registra todas las operaciones: `kubectl exec`, creación de pods, cambios de RBAC, acceso a secrets. Son la fuente de verdad del plano de control.

## Configuración de política de auditoría:

```
apiVersion: audit.k8s.io/v1
kind: Policy
rules:
  # Registrar todos los exec y port-forward (indicadores de intrusión):
  - level: Request
    resources: [{group: "", resources: ["pods/exec", "pods/portforward"]}]
  # Registrar acceso a secrets:
  - level: Metadata
    resources: [{group: "", resources: ["secrets"]}]
  # Registrar cambios de RBAC:
  - level: RequestResponse
    resources: [{group: "rbac.authorization.k8s.io"}]
```

## Indicadores de compromiso en audit logs:

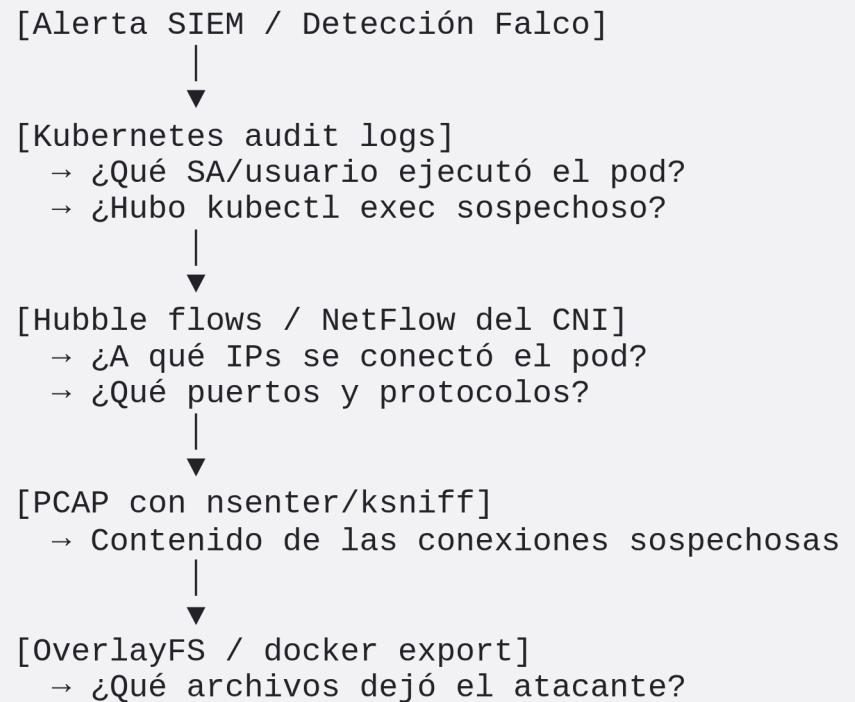
- ▶ \*\*`pods/exec`\*\* a pods en producción desde IPs no habituales
- ▶ Creación de \*\*ClusterRoleBinding\*\* privilegiado fuera de ventana de mantenimiento
- ▶ Acceso masivo a \*\*secrets\*\* desde una SA que normalmente no los lee
- ▶ \*\*`pods/portforward`\*\* a puertos internos — posible tunelización de tráfico

# Correlación entre capas — La visión completa

## Fuentes disponibles en entornos contenerizados:

- ▶ **Kubernetes audit logs** — quién hizo qué en el plano de control
- ▶ **Hubble/eBPF flows** — qué tráfico de red hubo entre pods
- ▶ **Falco alerts** — qué syscalls/comportamientos anómalos ocurrieron
- ▶ **Container runtime logs** — qué imagen, cuándo se inició, variables de entorno
- ▶ **OverlayFS / docker export** — qué archivos se crearon o modificaron
- ▶ **PCAP (nsenter/ksniff)** — tráfico de red en bruto si se capturó a tiempo

## Pipeline de correlación forense:



# Escenario práctico — Lateral movement en Kubernetes

**Escenario:** Se detecta tráfico inusual desde un pod del namespace `frontend`. El pod contacta con la API de Kubernetes y con pods del namespace `backend` en puertos no autorizados.

## Investigación paso a paso:

```
# 1. Identificar el pod y su service account:  
kubectl get pod <pod-sospechoso> -o yaml | grep serviceAccount  
  
# 2. Revisar qué permisos tiene esa SA:  
kubectl auth can-i --list --as=system:serviceaccount:frontend:default  
  
# 3. Ver audit logs del API server filtrando por esa SA:  
grep '"serviceAccount":"default"' /var/log/kube-apiserver-audit.log \  
| grep '"verb":"create\|exec\|list"' | jq .  
  
# 4. Capturar tráfico del pod con ksniff:  
kubectl sniff <pod-sospechoso> -n frontend -o /tmp/lateral.pcap  
  
# 5. Consultar flujos en Hubble:  
hubble observe --pod frontend/<pod-sospechoso> \  
--verdict FORWARDED --output json | jq '.destination'  
  
# 6. Extraer el sistema de archivos para análisis offline:  
docker export <container-id> | tar -x -C /mnt/evidencia/  
sha256sum /mnt/evidencia/**/* > /mnt/evidencia/_hashes.txt
```

# Resumen — Herramientas para forense en contenedores

## Captura de tráfico:

Herramienta	Método	Nivel
nsenter	Network namespace	L2-L4
ksniff	Plugin kubectl	L2-L7
Hubble	eBPF (Cilium)	L3-L7
Pixie	eBPF	L7 (HTTP/gRPC/DNS)

## Detección y alertas:

Herramienta	Enfoque
Falco	Syscalls y comportamiento
Tetragon	Procesos y red (eBPF)
Tracee	Forense de syscalls

## Análisis de sistema de archivos:

Comando	Uso
docker export	Extraer FS completo
docker cp	Copiar archivo concreto
docker inspect	Metadata del contenedor
crlctl inspect	containerd runtime info

## Plano de control:

Fuente	Información
K8s audit logs	Operaciones API server
RBAC audit	Escalada de privilegios
Envoy access logs	Tráfico mTLS en service mesh

## Regla de oro en contenedores:

La evidencia es **efímera** — capturar antes de que el pod desaparezca. Automatizar la respuesta con Falco + hooks de preservación de evidencia.

## A. Modelo TCP/IP

---

El análisis forense de red se centra principalmente en:

- ▶ **Capas 2, 3 y 4** del modelo OSI
- ▶ Las capas superiores se tratan generalmente como una única capa de aplicación
- ▶ Por eso se usa el **modelo TCP/IP** en lugar del OSI completo

**Ventaja:** Simplificación del modelo sin perder funcionalidad forense

## A. Modelo TCP/IP — Encapsulamiento

---

### Proceso de encapsulamiento:

1. **Aplicación**: Genera los datos de usuario
  2. **Transporte (TCP)**: Añade cabeceras
    - ▷ Puertos origen/destino, flags, número de secuencia
  3. **Red (IP)**: Añade cabeceras de enrutamiento
    - ▷ IPs origen/destino, TTL, protocolo
  4. **Enlace**: Añade cabeceras y cola (FCS)
    - ▷ MACs origen/destino, EtherType, CRC
- En destino**: Se desencapsula en sentido inverso

## A. Encapsulamiento OSI

---

---

Cada capa añade su propia información de control (PDU)

## A. Capas encapsuladas en Wireshark

---

### IMPORTANTE

Wireshark muestra su interpretación de la trama.

¡Esta interpretación puede no ser correcta!

Las capas están formadas por bytes que aparecen de forma secuencial en el paquete

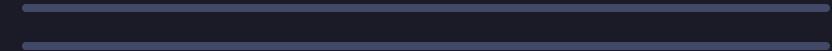
## A. Capas encapsuladas en Wireshark

---

### Visualización secuencial:

Las capas están formadas por bytes que aparecen de forma secuencial en el paquete

- ▶ **Datos HTTP**
- ▶ **Cabecera TCP**
- ▶ **Cabecera IP**
- ▶ **Cabecera Ethernet**



## A. Capas encapsuladas en Wireshark

---

### Estructura completa del paquete:

- ▶ Cada byte tiene su posición específica
- ▶ Los colores ayudan a identificar capas
- ▶ La interpretación depende del protocolo detectado
- ▶ Validar siempre en la vista hexadecimal

## B. Decimal — Binario — Hexadecimal

### Base 10 (Decimal)

- Dígitos del 0 al 9
- $198 = 1 \times 100 + 9 \times 10 + 8 \times 1$

### Base 2 (Binario)

- Dígitos 0 y 1
- $11000110 = 1 \times 128 + 1 \times 64 + 1 \times 4 + 1 \times 2 = 198$

### Base 16 (Hexadecimal)

- Dígitos del 0 al 9 y letras A-F
- $0xC6 = 12 \times 16 + 6 \times 1 = 198$

### Conversión Binario → Hex

1100 0110  
↓      ↓  
C      6

**Regla:** Cada 4 bits (nibble) = 1 dígito hexadecimal

**Fundamental** para leer cabeceras de protocolos en bruto

## B. Ejercicio — Cabecera UDP en Hex

### Ejercicio práctico:

Sea la cabecera UDP `0x0401 0035 004c 1fd7`

Calcular en decimal el puerto de origen y destino

### Solución:

- ▶ **Origen:**  $0401 \rightarrow 4 \times 256 + 0 \times 16 + 1 = 1025$
- ▶ **Destino:**  $0035 \rightarrow 3 \times 16 + 5 = 53$  (DNS)

Puerto 53 = DNS

Las primeras 4 palabras de 2 bytes de la cabecera UDP son siempre puerto origen, puerto destino, longitud y checksum

## C. Formato libpcap

---

### Cabecera del archivo (24 bytes):

- ▶ **Magic number:** 0xa1b2c3d4 (little-endian) o 0xd4c3b2a1 (big-endian)
- ▶ **Version major/minor:** versión del formato
- ▶ **Timezone offset:** siempre 0 (UTC)
- ▶ **Timestamp accuracy:** siempre 0
- ▶ **Snaplen:** máximo tamaño de paquete capturado
- ▶ **Link type:** tipo de enlace (1=Ethernet, 105=IEEE 802.11)

### Cada registro de paquete incluye:

- ▶ Timestamp (segundos + microsegundos)
- ▶ Longitud capturada vs longitud original

## C. pcap vs pcapng

### pcap (clásico)

- ▶ [X] Limitado a una interfaz
- ▶ [X] Metadata limitada
- ▶ [OK] Compatible universalmente
- ▶ [OK] Simple y rápido

### pcapng (moderno, desde 2004)

- ▶ [OK] Múltiples interfaces
- ▶ [OK] Comentarios y metadata por paquete
- ▶ [OK] Resoluciones de nombres embebidas
- ▶ [!] No todas las herramientas lo soportan aún

#### Conversión rápida:

```
# pcapng → pcap
editcap -F pcap \
    archivo.pcapng archivo.pcap

# pcap → pcapng
editcap -F pcapng \
    archivo.pcap archivo.pcapng
```

Wireshark guarda en pcapng por defecto — convertir si hay problemas de compatibilidad

## C. pcapng — Tipos de bloques

---

### Bloques en pcapng:

#### **Section Header Block (SHB)**

- ▷ Inicia cada sección, contiene metadatos del archivo

#### **Interface Description Block (IDB)**

- ▷ Describe cada interfaz de captura

#### **Enhanced Packet Block (EPB)**

- ▷ Contiene paquetes + timestamp + longitud + opciones

#### **Simple Packet Block (SPB)**

- ▷ Paquetes sin timestamp (más ligero)

#### **Name Resolution Block (NRB)**

- ▷ Mappings DNS/IP embebidos

#### **Interface Statistics Block (ISB)**

- ▷ Estadísticas de la interfaz al finalizar

La flexibilidad de bloques permite que una captura pcapng contenga tráfico de múltiples interfaces con diferentes link types en un único archivo

## D. Editcap — Editar archivos PCAP

### Editcap - Utilidad para editar archivos PCAP

- ▶ Dividir por rango de tiempo  
editcap -A "2025-01-01 00:00:00" -B "2025-01-02 00:00:00" \  
entrada.pcap salida.pcap
  - ▶ Dividir por número de paquetes  
editcap -c 1000 entrada.pcap salida.pcap
  - ▶ Ajustar timestamps (útil para corregir desfases NTP)  
editcap -t +3600 entrada.pcap salida.pcap
  - ▶ Crear subsets de paquetes por índice  
editcap entrada.pcap salida.pcap 1-100 200-300
  - ▶ Cambiar formato pcap
- ↔  
pcapng  
editcap -F pcap archivo.pcapng archivo.pcap
- ▶ Eliminar paquetes duplicados  
editcap -d entrada.pcap salida.pcap

## D. Mergecap y Capinfos

### Mergecap — Combinar PCAPs

```
# Combinar múltiples archivos  
mergecap -w salida.pcap \  
archivo1.pcap archivo2.pcap  
  
# Múltiples interfaces en pcapng  
mergecap -w salida.pcapng \  
eth0.pcap wlan0.pcap  
  
# Ordenar por timestamp  
mergecap -w salida.pcap \  
-F pcap *.pcap
```

Útil para consolidar capturas de múltiples sensores

### Capinfos — Información del PCAP

```
capinfos captura.pcap  
capinfos -T captura.pcap # tabla  
capinfos -l captura.pcap # largo
```

#### Muestra:

- ▶ Formato y encapsulación
- ▶ Número total de paquetes
- ▶ Timestamps inicio/fin y duración
- ▶ Bytes y bitrate promedio

Siempre ejecutar **capinfos** al recibir un PCAP para validar su integridad

## E. Capa de enlace — IEEE 802.x

---

### Conjunto de estándares del IEEE:

- ▶ **802.3** — Ethernet
- ▶ **802.11** — WiFi
- ▶ **802.15.1** — Bluetooth

### Ethernet II (el más común en redes corporativas):

- ▶ 14 bytes de cabecera + payload variable + 4 bytes CRC
- ▶ Campo **EtherType** identifica el protocolo L3:
  - ▶ IPv4: **0x0800** | IPv6: **0x86DD**
  - ▶ ARP: **0x0806** | VLAN 802.1Q: **0x8100**
- ▶ Tamaño mínimo de trama: 14 + 46 + 4 bytes

## E. Internet Protocol — Características

### Diseñado para:

- ▶ Manejar el **enrutamiento** y el **direcccionamiento** entre redes
- ▶ Operar en **capa 3** del modelo OSI

### Propiedades:

- ▶ **No orientado a conexión** — cada paquete es independiente
- ▶ **No confiable** — best-effort delivery
- ▶ Sin garantía de entrega, orden ni ausencia de duplicados
- ▶ La fiabilidad la aporta TCP en la capa superior

**Cabecera IP + Payload = Paquete IP**

Campos forenses clave:

- ▶ **TTL**: identifica el SO origen
  - ▷ Windows=128, Linux=64, Router=255
- ▶ **Protocol**: TCP=6, UDP=17, ICMP=1
- ▶ **Flags**: fragmentación DF/MF
- ▶ **ID**: identifica fragmentos del mismo datagrama

## Términos clave del curso

# Glosario (A–F)

---

## **ARP** (*Address Resolution Protocol*)

Protocolo que mapea una IP a su dirección MAC en la red local. Vulnerable a ARP spoofing.

## **Beaconing**

Comunicación periódica de un malware con su servidor C2. Patrón: intervalos regulares, pocos bytes.

## **BPF** (*Berkeley Packet Filter*)

Sintaxis de filtrado de paquetes usada por tcpdump y Wireshark. Opera a nivel de kernel.

## **C2** (*Command & Control*)

Infraestructura usada por el atacante para controlar el malware instalado en la víctima.

## **Cadena de custodia**

Registro documental de quién ha tenido acceso a una evidencia desde su recogida hasta el juicio.

## **DGA** (*Domain Generation Algorithm*)

Técnica de malware que genera dominios C2 automáticamente para dificultar el bloqueo.

## **ETA** (*Encrypted Traffic Analysis*)

Análisis del tráfico cifrado sin descifrarlo, usando metadatos como tamaño y timing.

## **Flow / NetFlow**

Resumen de una conversación de red: IPs, puertos, protocolo, bytes y paquetes. Menos detalle que PCAP pero mucho más ligero.

## **FPC** (*Full Packet Capture*)

Captura completa de todos los paquetes, incluyendo payload. Máximo detalle forense.

# Glosario (F-M)

---

## Fragmentación IP

División de un paquete IP en trozos más pequeños cuando supera el MTU. Puede usarse para evadir IDS.

## GDPR / RGPD

Reglamento General de Protección de Datos. Rige el tratamiento de datos personales en la UE, incluidos los capturados en tráfico de red.

## Hash (SHA-256)

Huella digital de un fichero. Se usa para verificar que una evidencia no ha sido alterada.

## IOC (*Indicator of Compromise*)

Señal de que un sistema ha sido comprometido: IP maliciosa, hash de malware, dominio C2.

## JA3 / JA4

Huella digital del TLS ClientHello. Permite identificar el cliente TLS sin descifrar el tráfico.

## Kill Chain

Modelo que describe las fases de un ataque: reconocimiento, entrega, explotación, instalación, C2, movimiento lateral, exfiltración.

## Lateral Movement

Técnica del atacante para moverse de un sistema a otro dentro de la red comprometida.

## Movimiento Norte-Sur

Tráfico entre la red interna y el exterior (Internet). Cruce del perímetro.

# Glosario (M–R)

---

## Movimiento Este-Oeste

Tráfico interno entre sistemas de la misma red.  
Más difícil de monitorizar que el Norte-Sur.

## mTLS (*Mutual TLS*)

Variante de TLS donde tanto cliente como servidor se autentican con certificados.

## PCAP (*Packet Capture*)

Fichero que contiene paquetes de red capturados. Formato más común: libpcap / pcapng.

## Perfect Forward Secrecy (PFS)

Propiedad de TLS 1.3: cada sesión usa claves efímeras. Impide descifrar capturas pasadas si se compromete la clave privada.

## Pivot

Movimiento del analista de una fuente de datos a otra siguiendo una pista: de alerta SIEM a PCAP en Arkime.

## Proxy

Intermediario entre cliente e Internet. Puede ser punto de captura y control de tráfico.

## QUIC / HTTP/3

Protocolo de transporte sobre UDP que reemplaza a TCP para HTTP/3. Dificulta el análisis forense clásico.

## REGEX (*Regular Expression*)

Patrón de búsqueda de texto. En Wireshark se usan con `matches` para detectar patrones en el payload.

## SPAN Port (*Switch Port Analyzer*)

Puerto de un switch configurado para recibir copias del tráfico de otros puertos. Alternativa económica al TAP.

# Glosario (S-Z)

---

## **SSLKEYLOGFILE**

Variable de entorno que hace que el navegador exporte las claves de sesión TLS. Permite descifrar PCAPs en Wireshark.

## **SYN scan / Port scan**

Técnica de reconocimiento que envía SYN a múltiples puertos para detectar servicios abiertos sin completar el handshake.

## **TAP (Test Access Point)**

Dispositivo hardware que intercepta y copia el tráfico de red de forma pasiva e indetectable.

## **Threat Hunting**

Búsqueda proactiva de amenazas usando hipótesis, antes de que se genere una alerta.

## **TLS (Transport Layer Security)**

Protocolo de cifrado de la capa de transporte. La versión 1.3 es el estándar actual.

## **TTL (*Time To Live*)**

Campo del paquete IP que indica cuántos saltos puede dar antes de descartarse. Útil para inferir el OS origen.

## **Wireshark**

Herramienta gráfica de análisis de paquetes de red. Estándar de facto en forense de red.

## **Zeek (antes Bro)**

Framework de análisis de red que genera logs estructurados por protocolo a partir de PCAP o tráfico en vivo.

## **Zero Trust**

Modelo de seguridad que no confía en ningún nodo por defecto, incluso dentro de la red corporativa. Implica cifrado ubicuo y microsegmentación.