WiFi traffic injection based attacks
Why all your WEP and open WiFi are belong to us

Cédric BLANCHER

cedric.blancher@eads.net
EADS Corporate Research Center
EADS/CCR/DCR/SSI

sid@rstack.org
Rstack Team
http://sid.rstack.org/

Pacsec/core05 - Tokyo - Japan
2005 November 15-16
http://pacsec.jp/
WiFi traffic injection based attacks
Quick spam...

<commercial>
EADS is a leading company in aeronautic, defense and space with products like A380 jetliner, Tigre helicopter or Ariane launcher

I’m part of Corporate Research Center IT Security Lab team in France.
</commercial>
Introduction

802.11 networks are well known to be vulnerable

- WEP is crippled
- Well-known LAN perimeter attack

So why this talk?
This talk is yet another "people never learn" story

Facts

- Most commercial hotspots rely on WiFi open networks
- 2/3 to 9/10 of networks are open or WEP networks
- Many WiFi capable devices only support WEP
- ISP providing WiFi capable wonder box only supporting WEP
WiFi traffic injection
WEP cracking
Bypassing captive portals
Attacking WiFi stations
WPA, WPA2 and 802.11i
Conclusion
References

Agenda

1 WiFi traffic injection
2 WEP cracking
3 Bypassing captive portals
4 Attacking WiFi stations
5 WPA, WPA2 and 802.11i
6 Conclusion
7 References
   • Demos
   • Bibliography
Introduction

Traffic injection basics
- Available chipsets and drivers
- How to inject and sniff
- Sample code example
On Linux, you can inject in monitor mode with:

- Prism2/2.5/3 with hostap[HAP] or wlan-ng[WLAN]
- Prism54 FullMAC with prism54[PR54]
- Atheros with madwifi[MADW]
- Ralink RT2x00 with rt2x00[RT2X]
- Realtek RTL8180 with rtl8180[RTL8]

Most drivers need patches written by Christophe Devine (see Aircrack[AIRC] tarball)
Frames injection and sniffing

You inject *and* sniff in monitor mode using the same adapter

```
# iwconfig ath0 mode monitor
# iwconfig ath0 channel 11
# ifconfig ath0 up promisc
```

You can read *and* write to ath0 directly\(^1\) with layer 2 socket (e.g. PF_PACKET)

\(^1\) Or purpose specific interface such as Madwifi ath0raw
Using Scapy[SCAP] as backend

```python
from scapy import Raw, Dot11, Dot11WEP, LLC, SNAP, sendp, conf
s = conf.L2listen(iface = "ath0")
conf.iface = "ath0"
```

Any 802.11 aware packet factory will do the trick...
Raw data frame injection

Send direct frame from SrcMAC to DstMAC

dot11_frame = Dot11(type = "Data",
                   FCfield = "to-DS",
                   addr1 = BSSID,
                   addr2 = SrcMAC,
                   addr3 = DstMAC)
dot11-frame /= LLC(ctrl=3)/SNAP()/"Raw data"
sendp(dot11_frame,verbose=0)
Reading date frames

Extract BSSID field value

dot11_frame = s.recv(1600)
if dot11_frame.getlayer(Dot11).FCfield & 1:
    BSSID = dot11_frame.getlayer(Dot11).addr1
else:
    BSSID = dot11_frame.getlayer(Dot11).addr2
Management traffic

Management traffic is easy to generate as well

- Dot11Disas
- Dot11AssoResp
- Dot11ReassoResp
- Dot11Deauth
- etc.
Agenda

1. WiFi traffic injection
2. WEP cracking
3. Bypassing captive portals
4. Attacking WiFi stations
5. WPA, WPA2 and 802.11i
6. Conclusion
7. References
   • Demos
   • Bibliography
WEP cracking

WEP basics

- RC4 cipher
- Auth with RC4
- CRC32 ICV
Attacks overview

Know attacks against WEP

- IV collisions
- Cleartext attacks (e.g. authentication challenge) and authentication bypass
- RC4 output/IV couple table construction
- Arbitrary frame injection
- Korek Chopchop attack
- Fluhrer, Mantin and Shamir attack (weak IVs attack)
- Korek optimization of FMS attack based on solved cases

Some of them can be boosted by traffic injection
"Your 802.11 Wireless Network Has No Clothes" [ASW01] WEP authentication is vulnerable to cleartext so you can grab 140 bytes of $RC4(IV \parallel K)$

**Challenge answer computation**

$$P' = (C' \parallel ICV(C')) \oplus RC4(IV \parallel K)$$

Once one authentication is captured, we can compute and inject any further answer $P'$ to challenge $C'$ using known RC4 output
RC4 output/IV tables

For every IV, grab RC4 output

- We know how to grab 140 bytes of RC4 output
- We can generate traffic with known RC4 output (e.g. GET / HTTP/1.0)
- We can have traffic generated and grab longer RC4 output (e.g. HTTP reply)

We can end up with a huge RC4 output/IV table (≈25GB) allowing one to decrypt any packet on the air
We can boost this attack playing with disassociations :)

Cédric BLANCHER
WiFi traffic injection based attacks
Modified frame injection

Let $C$ be our cleartext message and $C'$ a modification of $C$

Let $Mod = C \oplus C'$

**Arbitrary message modification**

\[
P = WEP(C \parallel ICV(C))
\]
\[
= (C \parallel ICV(C)) \oplus RC4(IV \parallel K)
\]

\[
P' = (C' \parallel ICV(C')) \oplus RC4(IV \parallel K)
\]
\[
= (C \parallel ICV(C)) \oplus RC4(IV \parallel K) \oplus (Mod \parallel ICV(Mod))
\]
\[
= P \oplus (Mod \parallel ICV(Mod))
\]

This means you can inject arbitrary layer 2 consistent WEP frames and have them decrypted...
Single packet inductive attacks

Arbaugh first published an inductive attack against WEP[ARB01]. Korek published a similar (reversed) inductive attack[KO04b] with a PoC called Chopchop.

1. Grab a multicast/broadcast frame
2. Strip the last data byte
3. Assume last byte cleartext value
4. Correct frame ICV and reinject
5. See if AP forwards the new frame

Extremely effective on ARP traffic (10-20s per packet).
Devine aircrack/aireplay WEP cracking

Christophe Devine wrote aircrack that relies FMS[FMS01] and Korek optimizations, and aireplay[AIRC] to inject traffic

1. Capture an ARP request, optionally checked with Chopchop
2. Inject ARP request again and again
3. Stimulate traffic and unique IV collection
4. Crack WEP key with optimized FMS

Full WEP cracking is now a matter of minutes[WACR]
And aircrack can still get optimized...
So WEP is weak...

Get the facts...

- Poll on Linux dedicated portal shows 80% users using open or WEP networks
- Recent study in "La Défense" business area near Paris shows 66% wardrivable non-hotspot accesses non protected
- 30 miles of wardriving in near Chicago shows 90% of 1114 accesses unprotected
- 21% use WPA (PSK or EAP)
WiFi traffic injection
WEP cracking
Bypassing captive portals
Attacking WiFi stations
WPA, WPA2 and 802.11i
Conclusion
References

1 WiFi traffic injection
2 WEP cracking
3 Bypassing captive portals
4 Attacking WiFi stations
5 WPA, WPA2 and 802.11i
6 Conclusion
7 References
   • Demos
   • Bibliography
Commercial public Internet access

- Captive portal based system
- Authentication to billing system through web portal
- Authorization for Internet access
- Authorization tracking
Once authenticated, users must be tracked

- MAC address
- IP address
- MAC and IP addresses

Those network parameters can easily be spoofed!
MAC based authorization tracking

Authorized clients are identified by their MAC address:
- MAC address is easy to spoof
- No MAC layer conflict on WiFi network
- Just need a different IP
MAC tracking bypass

Change WiFi interface MAC address

```bash
joker# ifconfig ath0 hw ether $MAC
joker# ifconfig ath0 $IP $NETMASK $BROADCAST
joker# route add default $FIREWALL
```
IP based authorization tracking

Authorized clients are identified by their IP address

- IP address are just a little more tricky to spoof
- ARP cache poisoning helps redirecting traffic
- Traffic redirection allows IP spoofing

See my LSM 2002 talk[BLA02], arp-sk website[ARPS] or MISC3[MISC] for details
"Smart spoofing"

```
joker# echo 1 > /proc/sys/net/ipv4/ip_forward
joker# arp-sk -i ath0 -w -d $FIREWALL -S $BATMAN \ 
   -D $FIREWALL
joker# iptables -t nat -A OUTPUT -d ! $LAN \ 
   -j SNAT --to $BATMAN
joker# iptables -t mangle -A FORWARD -d $BATMAN \ 
   -j TTL --ttl-inc 1
```
MAC+IP addresses based authorization tracking

The smart way for tracking people?
- Previous technic won’t help because of MAC address checking
- Send traffic with spoofed MAC address
- ARP cache poisoning and IP spoofing for answers redirection
Why does it work?

Layer2 and Layer3 are close to independant

- No correlation between ARP cache and filtering
- MAC spoofed frames are accepted
- Returning frames are sent with our MAC address
Reconfiguring the interface won’t help on this
We’ll use ebtables[EBT] to have output frames spoofed

joker# modprobe bridge
joker# brctl addbr br0; brctl addif br0 ath0
[configure bridge interface br0]
joker# ebtables -t nat -A POSTROUTING -o ath0 -d $FW_MAC \ 
   -j snat --to-source $BATMAN_MAC

Then you can apply IP spoofing and perform ”Smarter spoofing” :)
Few other technics

- Misconfigurations
- DNS based communication[OZY] or tunneling[NSTX]
- Administration network on the same VLAN, accessible through WiFi
- ESTABLISHED,RELATED -j ACCEPT prevents connections drop when authorization expires on Linux based systems
- Etc.
1. WiFi traffic injection
2. WEP cracking
3. Bypassing captive portals
4. Attacking WiFi stations
5. WPA, WPA2 and 802.11i
6. Conclusion
7. References
   - Demos
   - Bibliography
Associated stations are almost naked

- LAN attacks (ARP, DHCP, DNS, etc.)
- Traffic interception and tampering
- Direct station attacks

Remember the infamous personal firewalls exception for local network...
Traffic tampering with injection

WiFi communication can be listened on the air
- Listen to WiFi traffic
- Catch interesting requests
- Spoof AP and inject your own answers
- Clap clap, you’ve done airpwn-like[AIRP] tool

Only think of injecting nasty stuff in HTTP traffic, just in case someone would dare to use MSIE on an open WLAN
Station to station traffic prevention

Security feature that blocks traffic within BSS
Usually known as *station isolation*
- Station sends To-DS frame
- AP sees destination is in BSS
- AP drops the frame

No From-DS frame, so no communication
d: stations can’t talk to each other...

\[a\]Does not work between 2 APs linked via wired network
Isolation bypass using traffic injection

Joker can inject From-DS frames directly

- No need for AP approval
- You can spoof about anyone
- You’re still able to sniff traffic

Traffic injection allows complete isolation bypass
Full communication with injection

Sending traffic directly to stations allows direct station to station communication, even if:

- AP applies restrictions
- AP refuses association
- AP is out of reach

A smart way for talking to stations without being associated
Attacking stations
Proof of concept: Wifitap

Needed a PoC for Cisco PSPF bypass and wrote Wifitap
- Written in Python
- Relies on Scapy
- Uses tuntap device and OS IP stack
- Use WiFi frame injection and sniffing

Wifitap allows communication with station despite of AP restrictions
Wifitap usage

# ./wifitap.py -h
Usage: wifitap -b <BSSID> [-o <iface>] [-i <iface> [-p]]
[-w <WEP key> [-k <key id>]]
[-d [-v]] [-h]

-b <BSSID> specify BSSID for injection
-o <iface> specify interface for injection
-i <iface> specify interface for listening
-p No Prism Headers in capture
-w <key> WEP mode and key
-k <key id> WEP key id (default: 0)
-d activate debug
-v verbose debugging
-h this so helpful output
## Wifitap in short

### How Wifitap works

<table>
<thead>
<tr>
<th>Sending traffic</th>
<th>Receiving traffic</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Read ethernet from tuntap</td>
<td>- Sniff 802.11 frame</td>
</tr>
<tr>
<td>- Add 802.11 headers</td>
<td>- Remove WEP ifd needed and 802.11</td>
</tr>
<tr>
<td>- Set BSSID, From-DS and WEP if needed</td>
<td>- Build ethernet frame</td>
</tr>
<tr>
<td>- Inject frame over WiFi</td>
<td>- Send frame through tuntap</td>
</tr>
</tbody>
</table>

Attacker does not need to be associated
Hotspots with isolation

Some hotspots implement isolation to prevent clients from attacking each other

- Does not protect against "session" hijacking
- Attacker must then to take over victim’s session
- Victim does not have access anymore, and still pays for it

And among all, it’s pretty useless...
Hijacking people authorization is not very kind

- Use Wifitap to bypass isolation
- Now you can route back his traffic to your victim

Your victim and you are both able to surf transparently

Now, you ”can be a true gentlemanly [h|cr]acker”[ISCD] ;)

More hotspot bypassing...
Agenda

1. WiFi traffic injection
2. WEP cracking
3. Bypassing captive portals
4. Attacking WiFi stations
5. WPA, WPA2 and 802.11i
6. Conclusion
7. References
   - Demos
   - Bibliography
WPA

Transitional recommendation[WPA] from WiFi Alliance (2003) extracted from IEEE work for infrastructure networks only

- New authentication scheme based on PSK or 802.1x
- New key generation and scheduling scheme for keys
- New integrity check through Michael MIC with sequencing

Pretty solid solution that can prevent injection/replay
WPA2 and 802.11i

802.11i[IEEE04b] is a standard from IEEE for WiFi security. WPA2[WPA2] is a recommendation from WiFi Alliance based on 802.11i.

- RSN\(^2\) concept: security algorithms negotiation
- Integrates Ad-Hoc security
- Authentication using 802.1x
- Ciphering using AES-CCMP
- Integrity check using CCMP MIC

Return to the roots and use of a real adapted ciphering solution

\(^2\)Robust Security Network
Some flaws already

Yet some papers have been published regarding WPA/WPA2 security

- WPA weak PSK (<20 chars) bruteforce[MOS03]
- Injection of spoofed first RSN handshake message leads to memory exhaustion[HM04] (DOS)
- TEK attack in $2^{105}$ instead of $2^{128}$ (requires key knowledge)[MRH04] on TKIP
- Counter-measures abuse (DOS) : traffic replay, dumb traffic injection

Moreover, nothing will ever protect from layer 1 based DoS attacks (bandwidth reservation, jamming)
Setting up WPA/WPA2

Building WPA/WPA2 aware network

**Client side**
- Windows 2000SP4
- MacOS 10.3 Panther
- Linux/BSD with wpa_supplicant[WPAS]

**Access Point side**
- All APs since 2003
- Upgrade firmware!
- Linux/BSD with hostapd[HAPD]
And then?

Although some flaws, WPA provides strong mechanisms for end users

- Good authentication mechanisms if properly used
- Real per-user session management
- Session key management and re-keying
- Real integrity check
- Anti-replay, anti-injection mechanisms

WPA2 is even better with AES-CCMP support.
WiFi traffic injection
WEP cracking
Bypassing captive portals
Attacking WiFi stations
WPA, WPA2 and 802.11i

Conclusion
References

1 WiFi traffic injection
2 WEP cracking
3 Bypassing captive portals
4 Attacking WiFi stations
5 WPA, WPA2 and 802.11i

6 Conclusion

7 References
   - Demos
   - Bibliography
Then...

- Don’t use WEP anymore, it ”has no clothes” at all
- Don’t use open networks for public access, use WPA/WPA2\(^a\)
- Migrate to WPA, then WPA2 as soon as possible

\(^a\)BTW, RADIUS is far better for AAA

Vendors, journalists, etc. : stop telling people WEP is OK
Manufacturers : provide WPA/WPA2 support out of the box
Maybe deprecating WEP support could help (or not)?
Thank you for your attention and...

Greetings to...

- EADS CCR/DCR/STI/C team
- **Rstack.org** team  
  http://www.rstack.org/
- **MISC Magazine**  
  http://www.miscmag.com/
- **French Honeynet Project**  
  http://www.frenchhoneynet.org/

Download theses slides from http://sid.rstack.org/
WiFi traffic injection
WEP cracking
Bypassing captive portals
Attacking WiFi stations
WPA, WPA2 and 802.11i
Conclusion
References

Demos
Bibliography
WiFi traffic injection

WEP cracking

Bypassing captive portals

Attacking WiFi stations

WPA, WPA2 and 802.11i

Conclusion

References

Demos

Bibliography
Demos

- WEP cracking
- WiFi traffic tampering
- WiFi traffic injection based communication
- Captive portal bypass
1. WiFi traffic injection
2. WEP cracking
3. Bypassing captive portals
4. Attacking WiFi stations
5. WPA, WPA2 and 802.11i
6. Conclusion
7. References
   - Demos
   - Bibliography


Bibliography II


[RW95] A. Roos and D.A. Wagner, Weak keys in RC4, sci.crypt Usenet newsgroup


Bibliography III


WiFi traffic injection
WEP cracking
Bypassing captive portals
Attacking WiFi stations
WPA, WPA2 and 802.11i
Conclusion
References

Bibliography IV


[ARB01] W.A. Arbaugh, An Inductive Chosen Plaintext Attack against WEP/WEP2, 2001,
http://www.cs.umd.edu/~waa/attack/v3dcmnt.htm

[BLA02] C. Blancher, Switched environments security, a fairy tale, 2002,
http://sid.rstack.org/pres/0207_LSM02_ARP.pdf

http://sid.rstack.org/pres/0307_LSM03_L2_Filter.pdf

[KO04a] Korek,
Bibliography VI

[KO04b] Korek, Chopchop,
http://www.netstumbler.org/showthread.php?t=12489

[AIRC] C. Devine, Aircrack,
http://www.cr0.net:8040/code/network/aircrack/


Bibliography VII

[HAPD] Hostapd authenticator,
http://hostap.epitest.fi/hostapd/

[MADW] MadWiFi project,
http://madwifi.sourceforge.net/


[OZY] OzymanDNS,
http://www.doxpara.com/ozymandns_src_0.1.tgz

[PR54] Prism54 Linux driver, http://prism54.org/

Bibliography VIII


[ISCD] ISC Handler’s Diary,