



# Wi-Fi Advanced Stealth

BlackHat US, Las Vegas – August 2-3, 2006

Laurent BUTTI & Franck VEYSSET  
France Telecom Division R&D

*firstname dot lastname AT orange-ft dot com*





# Who Are We?

- Network security experts in R&D labs
  - Working for France Telecom - Orange (major telco)
- Speakers at security-focused conferences
  - ShmooCon, ToorCon, FIRST, Eurosec...
- Wi-Fi security focused speakers ;-)
  - “Wi-Fi Security: What’s Next” – ToorCon 2003
  - “Design and Implementation of a Wireless IDS” – ToorCon 2004 and ShmooCon 2005
  - “Wi-Fi Trickery, or How To Secure (?), Break (??) and Have Fun With Wi-Fi” – ShmooCon 2006





# Beginning of 2006...

- We released 3 new tools at ShmooCon 2006
  - Raw Fake AP: an enhanced Fake AP tool using **RAW** injection for increased effectiveness
  - Raw Glue AP: a Virtual AP catching every client in a virtual quarantine area
  - Raw Covert: a tricky 802.11 covert channel using valid **ACK** frames
- All this stuff is available at
  - <http://rfakeap.tuxfamily.org>





# Now at BlackHat US...

- We will release
  - Tricks to “hide” access points and stations (madwifi patches)
    - From scanners and wireless IDS
  - Raw Covert v2: new implementation and features
- We will also introduce our new ideas of research
  - 802.11 fuzzing





# Wi-Fi Stealth Tricks





# 802.11 Havoc!

- Since a couple of years, some wireless drivers are much more “flexible” than Prism2/2.5/3 based...
  - Full **RAW** injection capabilities (possible to modify some critical fields like fragmentation, sequence number, BSS Timestamp...)
    - Demonstrated by Raw Fake AP, Raw Glue AP and Raw Covert
  - Tweaking the driver may also become attractive!
- Such drivers are
  - Madwifi-~~old~~ng for Atheros chipsets
  - Prism54.org for Prism54 chipsets
  - Realtek...
- New capabilities implies new risks to address...
  - Especially for Wireless IDS vendors





# Two Ways To Achieve Stealth...

- Possibilities are somewhat infinite...
  - We decided to show only two ways that can be extended
- Tweaks in 802.11 drivers to implement a new “proprietary” protocol over 802.11 bands
  - Madwifi patches
- Covert channel using 802.11 valid frames
  - Raw Covert (as a proof-of-concept)





# Hiding Ourselves







# A Quick Reminder

- IEEE 802.11 standards define what 802.11 is
  - At PHY and MAC layers
    - Modulation, frequencies...
    - State machine, frame fields...
    - Security mechanisms
- To be Wi-Fi compliant, every implementation must comply with the 802.11 standard and be certified by the Wi-Fi Alliance certification process
  - Usual stuff if you want to interoperate...





# Main Idea

- What would happen if you implement your own 802.11 stack?!
  - Stations that probe for AP will (probably) not see you...
  - Wireless sniffers will (probably) not understand you, requiring manual inspection...
  - Wireless IDS will (probably) not see you...
- Quite stealthy, no?
- What about your own (undetected) personal AP?
  - Sure the CSO won't appreciate 😊
  - Sure wardrivers won't appreciate either (until now...)





# Implementation

- Successfully tested on Atheros chipsets with a patched **madwifi-ng** driver
  - Patched stations and access points will be able to see and associate themselves (they speak the same language)
  - But non patched stations will not see patched access points, and thus cannot associate to them
- Test bed
  - Windows XP supplicant and NetStumbler
  - Wireless Tools (**iwlist**) with
    - **hostap**, (non patched) **madwifi-ng**, **ipw2100**, **prism54**





# Live Demonstration

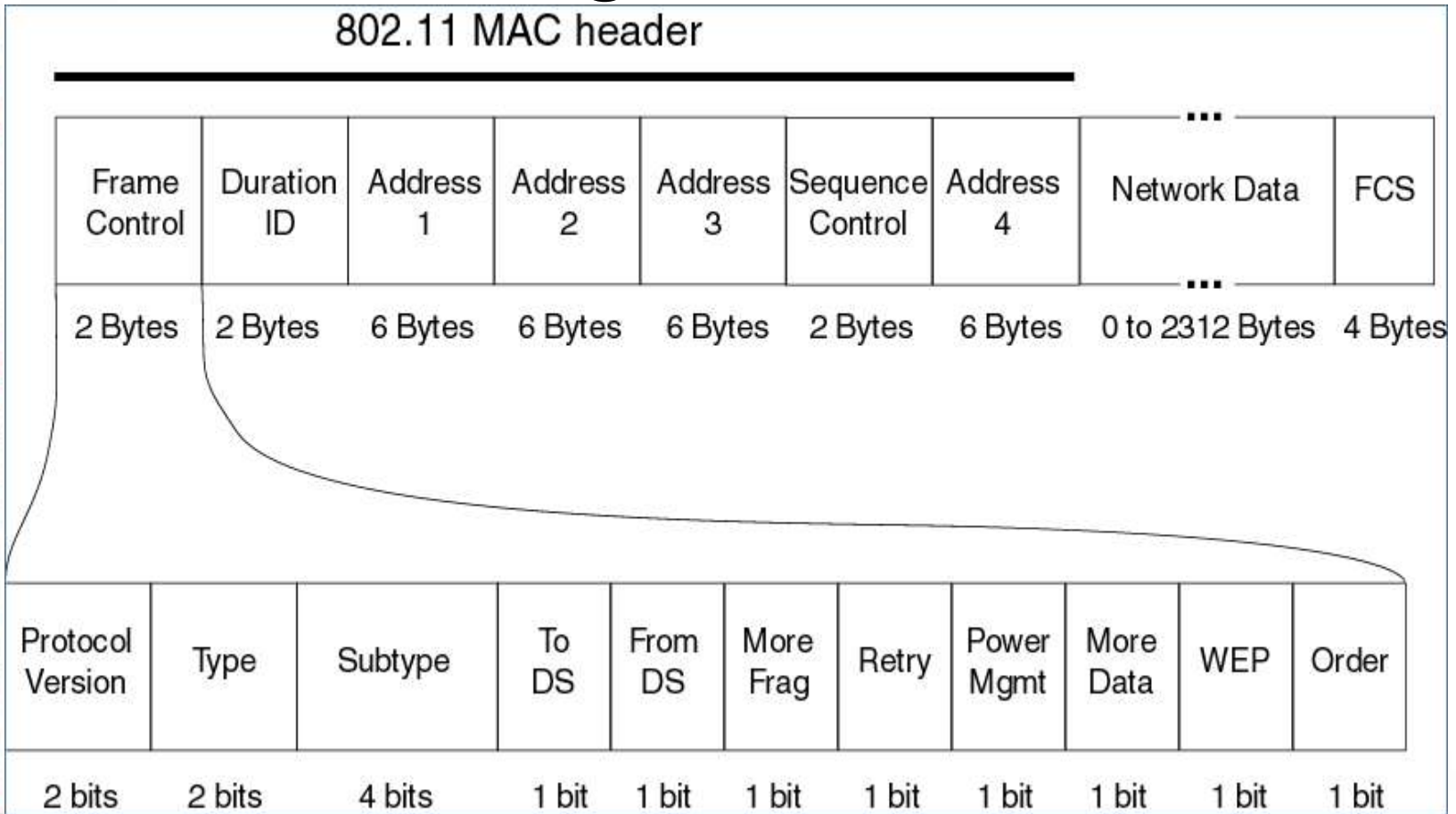
- First, we set up a “special” Access Point
  - one laptop with a patched `madwifi-ng` in master mode
- Then we scan for this AP with unpatched `madwifi-ng`
  - `iwlist` (active scan facilities under \*nix)
  - Kismet (passive scanner under \*nix)
  - Netstumbler (active scanner under Windows)
- Then, we use our “special” client (patched drivers)
  - Tada... it works...





# Design Details

802.11 MAC header





# WTF Is This? Trivial Tweaks!

- What about changing FC field? ;-)
- What about a protocol version of 1? ;-)
  - 802.11 is protocol version 0
- What about swapping types?
  - Management (value 0)
  - Control (value 1)
  - Data (value 2)
  - Reserved (value 3)
- What about swapping subtypes?
  - Is this a Probe Request or a Probe Response? ;-)





# Not So Trivial Tweaks

- Everything is possible... Make your own MAC protocol
- SoftMAC: A Flexible Wireless Research Platform
  - <http://systems.cs.colorado.edu/projects/softmac>





# Proto Tweak (1<>0)

Chipset	Driver	iwlist	Netstumbler
Prism54	Prism54 1.2	Not detected	Not tested
Prism2.5	Hostap 0.4.4	Not detected	Not tested
Atheros ar5212	Madwifi-ng r1527	Not detected	Not tested
Atheros ar5211	2.4.1.30 (win)	Not detected	Not detected
Centrino 2100	lpw2100 1.1.3	Not detected	Not tested
Atheros	Madwifi-ng patched	OK !	Not tested







# About Kismet

- Kismet runs in monitor mode
  - Will spot some of our AP
  - ...it depends on the tweak
  - Or will report high « Discrd » packets number 😊





No. .	Time	Source	Destination	Protocol	Info
1	0.000000	00:0d:28:3e:a0:5a	ff:ff:ff:ff:ff:ff	IEEE 802	Beacon frame, SN=133, FN=0, BI=100, SSID: "FTRDWPA"
2	0.066695			IEEE 802	Unrecognized (Reserved frame)
3	0.102364	00:0d:28:3e:a0:5a	ff:ff:ff:ff:ff:ff	IEEE 802	Beacon frame, SN=134, FN=0, BI=100, SSID: "FTRDWPA"
4	0.169095			IEEE 802	Unrecognized (Reserved frame)
5	0.204761	00:0d:28:3e:a0:5a	ff:ff:ff:ff:ff:ff	IEEE 802	Beacon frame, SN=135, FN=0, BI=100, SSID: "FTRDWPA"
6	0.215001			IEEE 802	Unrecognized (Reserved frame)
7	0.215303		00:14:6c:53:17:b6 (RA)	IEEE 802	Acknowledgement
8	0.216233			IEEE 802	Unrecognized (Reserved frame)
9	0.216549		00:11:0a:80:27:c9 (RA)	IEEE 802	Acknowledgement

Frame 4 (118 bytes on wire, 118 bytes captured)

Arrival Time: Jul 12, 2006 09:23:36.953879000  
[Time delta from previous packet: 0.066731000 seconds]  
[Time since reference or first frame: 0.169095000 seconds]  
Frame Number: 4  
Packet Length: 118 bytes  
Capture Length: 118 bytes  
[Protocols in frame: wlan]

IEEE 802.11

Type/Subtype: Unknown (56)

Frame Control: 0x008C (Normal)

Version: 0  
Type: Unknown (3)  
Subtype: 8

```
0000 8c 00 00 00 ff ff ff ff 00 11 0a 80 27 c9 .....
0010 00 11 0a 80 27 c9 00 f0 81 11 58 47 00 00 00 00 .....XG....
0020 64 00 21 05 00 04 61 74 68 31 01 08 82 84 8b 96 |d|...athl....
0030 0c 12 18 24 03 01 01 05 04 00 01 00 00 07 06 4e |.|$.....N...
0040 41 49 01 0e 12 20 01 00 2a 01 00 32 04 30 48 60 |A|...*.2.OH...
0050 6c dd 18 00 50 f2 02 01 01 82 00 02 a3 40 00 27 |.|P.....@'...
0060 a4 00 00 42 43 5e 00 62 32 2f 00 dd 09 00 03 7f |.|BC^b 2/.....
0070 01 01 00 24 ff 7f .....$..
```





# Raw Covert





# Raw Covert (1/4)

- Covert channel
  - In information theory, a covert channel is a communications channel that does a writing-between-the-lines form of communication.
  - Source: Wikipedia, the free encyclopedia
- Writing between-the-lines
  - Use valid frames to carry additional information
  - Valid frames could be management, control or data frames
- This tool is 'only' an example! Possibilities are infinite!





# Raw Covert (2/4)

- With 802.11, this may be performed by many means
  - Using a proprietary protocol within valid or invalid frames
  - It gives infinite possibilities thanks to **RAW** injection
- (Some) 802.11 frames are not considered as ‘malicious’
  - Control frames like **ACK** are lightweight and non suspicious!
    - Frame control (16 bits)
    - Duration Field (16 bits)
    - Receiver Address (48 bits)
  - (Usually) not analyzed by wireless IDS
    - No source nor BSSID addresses ;-)
- (Some) 802.11 drivers do not give back **ACK** frames in monitor mode (operated in the firmware: e.g. HostAP)
  - Increasing stealthiness





# Raw Covert (3/4)

- How it works?
  - A client encodes the information and sends **ACKs** over the air
  - A server listens for **ACKs** and tries to decode the information
- Basically, it uses a magic number in receiver address
  - 2 bytes
- Basically, it encodes the covert channel in receiver address
  - E.g. 4 bytes
- Several **ACK** frames are needed to send information





# Raw Covert (4/4)

- Issues
  - **ACK** frames can be missed, wireless is not a reliable medium! ;-)
  - Detection may be performed (only) with anomaly detection
- Enhancements
  - Basic remote shell and file transfer
  - Tun/tap interface
- Possible enhancements for the covert channel
  - Using invalid frames
  - Using Information Elements in 802.11 frames (but could be easily detected)
  - Using existing communications (clients and access points)





# Raw Covert Enhancements (1/2)

- Invalid frames (in the 802.11 sense, i.e. proprietary frames)
  - But would (?) be detected by any wireless IDS performing sanity check on every frame
- FCS invalid frames
  - Should require driver/firmware modifications to inject bad FCS
  - Wireless IDSs do not analyze such bad frames
  - But should be detected with FCSerr statistics (even if harder to diagnose as a covert channel)







# Raw Covert Enhancements (2/2)

- Invalid FCS monitoring
  - Usually a bit is set by the firmware when a FCS is invalid
  - Most drivers discard packets with bad FCS thanks to this information
    - `HAL_RXERR_CRC` for madwifi
    - `rfmon_header->flags & 0x01` for prism54
  - HostAP driver has a facility
    - `prism2_param interface monitor_allow_fcser 1`





# Live Demonstration

- Live demo!
- Did you detect it? ;-)





Filter: wlan.fc.type\_subtype == 29 + Expression... Clear

No. ↓	Time	Source	Destination	Protocol	Info
277	13.729472		01:02:00:2a:00:00 (RA)	IEEE 802	Acknowledgement
281	13.729844		03:04:ff:ff:ff:ff (RA)	IEEE 802	Acknowledgement
283	13.730136		03:04:ff:ff:ba:c2 (RA)	IEEE 802	Acknowledgement
285	13.730345		03:04:70:cc:18:de (RA)	IEEE 802	Acknowledgement
287	13.730552		03:04:08:06:00:01 (RA)	IEEE 802	Acknowledgement
289	13.730861		03:04:08:00:06:04 (RA)	IEEE 802	Acknowledgement
291	13.731171		03:04:00:01:ba:c2 (RA)	IEEE 802	Acknowledgement
293	13.731481		03:04:70:cc:18:de (RA)	IEEE 802	Acknowledgement
295	13.731789		03:04:01:01:01:01 (RA)	IEEE 802	Acknowledgement
296	13.732100		03:04:00:00:00:00 (RA)	IEEE 802	Acknowledgement
297	13.732410		03:04:00:00:01:01 (RA)	IEEE 802	Acknowledgement
298	13.732864		03:04:01:02:00:00 (RA)	IEEE 802	Acknowledgement
300	13.734388		05:06:00:00:00:00 (RA)	IEEE 802	Acknowledgement
324	14.761486		01:02:00:2a:00:00 (RA)	IEEE 802	Acknowledgement
327	14.761964		03:04:ff:ff:ff:ff (RA)	IEEE 802	Acknowledgement
329	14.762225		03:04:ff:ff:ba:c2 (RA)	IEEE 802	Acknowledgement
331	14.762546		03:04:70:cc:18:de (RA)	IEEE 802	Acknowledgement
333	14.762863		03:04:08:06:00:01 (RA)	IEEE 802	Acknowledgement

- ▶ Frame 277 (154 bytes on wire, 154 bytes captured)
- ▶ Prism Monitoring Header
- ▶ IEEE 802.11





# 802.11 Fuzzing





# Fuzzing Concepts (1/2)

- Fuzzing
  - Fuzz testing is a software testing technique. The basic idea is to attach the inputs of a program to a source of random data. If the program fails (for example, by crashing, or by failing built-in code assertions), then there are defects to correct.
  - From Wikipedia, the free encyclopedia





# Fuzzing Concepts (2/2)

- Fuzzing is not something really new...
  - Remember ISIC?
    - <http://www.packetfactory.net/projects/ISIC/>
- But it is still of interest...
  - Recent work on Bluetooth Fuzzing (Pierre Betouin)
    - <http://www.secuobs.com/bss-0.6.tar.gz>
  - Fuzzing with Scapy... (Phil Biondi)
    - Plenty of cool things to do with scapy...





# Fuzzing 802.11

- IEEE 802.11 amendments are more and more numerous
  - 802.11e, 802.11i, 802.11k, 802.11r, 802.11s, 802.11w...
- Axiom
  - Complexity → more code → more bugs → more vulnerabilities
- Guess what? IEEE 802.11 may be susceptible to fuzzing!





# Fuzzing 802.11

- Not so trivial... keep in mind the 802.11 state machine
- Each step of the 802.11 protocol may be fuzzed
  - Scanning process: probe requests and responses, beacons
  - Authentication process: authentication requests and responses
  - (Re-)Association process: (re-)association requests and responses
- Station's associated state can be fuzzed only if
  - Station is in state « Authenticated, Not Associated »
  - (Optionally) There was an (re-)association request sent by the station to the access point were he was previously authenticated







# Fuzzing 802.11

- Easiest part: fuzzing clients thanks to probe responses and beacons
  - Listen for probe requests and send back appropriate probe response
- Fuzzing probe responses and beacons
  - Inconsistent Information Elements (Type Length Value)
    - E.g. a SSID Information Element with a length above 32 bytes
    - E.g. a short 802.11 frame (incomplete SSID IE)
  - Incomplete frame length...
- More on this soon...





Thanks for your attention

*Tools, patches available at*  
<http://rfakeap.tuxfamily.org>





# References

- Laurent Oudot's **wknock**
  - <http://www.rstack.org/oudot/wknock/>
- Pierre Betouin's **Bluetooth Stack Smasher**
  - <http://www.secuobs.com/bss-0.6.tar.gz>
- **scapy** (Phil Biondi)
  - <http://www.secdev.org>
- **SoftMAC: A Flexible Wireless Research Platform**
  - <http://systems.cs.colorado.edu/projects/softmac>
- **MadWiFi patches and rawcovert**
  - <http://rfakeap.tuxfamily.org>

