High Speed Risks in 802.11n Networks

Joshua Wright
Aruba Networks
4/17/08 | WIR-301
Introduction

- IEEE 802.11n technology introduction
- Availability risks for legacy networks
- Extended range in 802.11n
- 40 MHz monitoring challenges
- Evading WIDS rogue detection systems
- Built-in DoS vulnerability
- Driver flaws
Disruptive Changes to Access Layer

• 802.11n promises to revolutionize the deployment of the network access layer

• Many organizations considering 802.11n as a wired replacement for new LAN deployments
  • Cost benefits, application integration benefits

• Represents a viable mechanism for reliability, consistency in connectivity and performance
  • Not without its own costs …
  • Not without its own risks …
IEEE 802.11n Overview

• TGn working group goal to improve PHY and MAC layers for true 100 Mbps performance

• PHY layer features include MIMO, 40 MHz channel availability

• MAC layer features include data aggregation, block acknowledgement

• Currently shipping hardware based on 802.11n D2.0, D4.0 currently in editing process

• Approval tentatively scheduled for 7/2009
New Spectrum Utilization

• 802.11 networks use 22/20 MHz channels
• Channel numbers separated by 5 MHz (mostly)
  • Channel 1 is 2.412 GHz, channel 2 is 2.417 GHz
  • Channel 44 is 5.220 GHz, channel 48 is 5.240 GHz
• Common 2.4 GHz deployments on 1, 6 and 11
• 802.11n can use 20 and 40 MHz channels
  • e.x. Channel 44 and 48 used together for more bandwidth
• Becomes problematic for 2.4 GHz band
2.4 GHz band, 40 MHz channels

• 40 MHz channels works well at 5 GHz

• 40 MHz channels at 2.4 GHz are problematic due to how channels are utilized
  • Channel 1 at 40 MHz utilizes 2.402 GHz - 2.442 GHz
  • Overlaps with channels 1 - 7
  • Leaves only one remaining viable channel

• Coordination effort flawed at 2.4 GHz, 40 MHz channels do not align with channels 1, 6, 11

• WFA requires 40 MHz @ 2.4 GHz off by default
Range Extensions in 802.11n

• SISO transmitters suffer from multipath propagation, reduces effective transmit distance
• MIMO transmitters leverage multipath to transmit multiple signals simultaneously
• Effectively increases range of 802.11n networks
• Legacy client support requires traditional site-survey planning

MIMO planning should expect 1.5x to 4x the range of SISO networks
Standard 802.11a/g deployment range estimate
MIMO upgrade using existing AP locations
WIDS Channel Monitoring

• Overlay WIDS systems utilize channel hopping to monitor all frequencies
  • Necessary to identify attacks on channels not utilized
• With more channel availability, WIDS sensor spends less time on each channel
• If sensor spends 1/10th second on each channel, attack has to last for ~4 seconds to be detected
40 MHz WIDS Monitoring

- Each channel must be monitored at 20 and 40 MHz
- Attack has to last for ~8 seconds to be detected
Evading WIDS Systems

• High Throughput (HT) mixed mode designed to be backward-compatible with existing chips
  • Performance degradation similar to 802.11b/802.11g

• Maximum 802.11n performance achieved through HT greenfield format
  • Not backward compatible with existing cards

• Legacy 802.11a/b/g WIDS systems unable to decode greenfield mode data
Evading Rogue Detection Mechanisms

- Rogue: unauthorized AP on your network
  - Essentially "Ethernet jack in your parking lot"

- WIDS systems significant value-add is identification and IPS against rogue APs

- Greenfield rogue devices can be used to evade existing WIDS analysis systems
  - Cannot be detected by WIDS without 802.11n card
  - Allows "attacker" to evade policy and enforcement mechanisms
Built-In DoS Vulnerability

• Positive acknowledgement of all data frames

• Real-time applications may not need positive acknowledgement

• Block ACK introduced in 802.11e, enhanced in 802.11n D3.0
  • Receiver positively or negatively acknowledge multiple frames within a negotiated window
  • 802.11 sequence numbers used for identification

• Enhanced in 802.11n for frame aggregation
Identify Starting SN, Window of Seq#s

Transmit multiple frames

Request Packet RX Report

Report on RX frames
Block Acknowledgement Handling

- ADDBA Request, transmitter specifies start and end of sequence numbers receiver should expect
  - WinStart_B is the next expected sequence number
  - WinSize_B is the block size of sequence numbers
  - WinEnd_B = (WinStart_B + WinSize_B) - 1

- Receiver accepts frames within the window
  - WinStart_B <= SN <= WinEnd_B
  - Frames outside of window are dropped, cannot be acknowledged with block ACK
Vulnerability in Block ACK Handling

• Recipient receives or drops frames according to WinStart_B and WinEnd_B values

• Attacker can impersonate ADDBA frames
  • Control frame, no security applied

• Artificially modifying WinStart_B and WinEnd_B causes all other frames to be dropped

• BlockAckReq "status report" will indicate multiple frames missed
I received 411, 412

My new window is 2460..2542

All dropped, outside of window

I received 411, 412

Originator

| 411 | QoS Data |
| 412 | QoS Data |
| 2460 | ADDBA Request |
| 413 - 432 | QoS Data |
|  | ... |
|  | BlockAckReq |
|  | BlockAck |
| ?? | ?? |

Recipient

Window (411..432)
802.11n Block ACK DoS Vulnerability

• All traffic is discarded until the new window is reached
• Transmitting station gets TX report, knows that frames were not received
  • Could be retransmitted, if buffered
  • Defeats the purpose of block acknowledgement
  • Impact will be implementation-dependent
• Attacker can repeat with new ADDBA messages to keep moving valid window
• No plans to address in 802.11n
• Sometimes DoS vulnerabilities are considered acceptable
Driver Flaws

• Next-generation attack vehicle for 802.11 networks

• Attackers recognize strength of WPA/WPA2 with AES-CCMP and EAP/TLS or PEAP/TTLS

• Attackers migrating to exploiting client vulnerabilities
  • Crafting malformed frames that trigger software vulnerabilities on a target machine
  • Executed with few packets, full compromise of target
Discovery: 802.11 Protocol Fuzzing

• Protocol fuzzing sends malformed input to test for programming flaws, bugs
• Identified flaws often turn into buffer/heap overflow vulnerabilities
• Flaws exploited by attackers at layer 2
• Little protection from firewalls at layer 3
• Recent public attention at hacker conferences, academic publications, commercial tools
The length of the SSID information field is between 0 and 32 octets. A 0 length information field indicates the broadcast SSID." IEEE 802.11-1999 p 55
#!/usr/bin/python
import sys
from scapy import *
target = "00:09:5B:64:6F:23"
ap = "00:40:96:01:02:03"
conf.iface = "wlan0"
basep = Dot11(
    proto=0, type=0, subtype=5,               # Probe response frame
    addr1=target, addr2=ap, addr3=ap,        # sent to target from AP
    FCfield=0, SC=0, ID=0)                   # other fields set to 0
basep /= Dot11ProbeResp(
    timestamp = random.getrandbits(64),     # Random BSS timestamp
    beacon_interval = socket.ntohs(0x64),   # byte-swap BI, ~.10 sec
    cap = socket.ntohs(0x31))                # AP/WEP/Short Preamble
ssid = "fuzzproberesp"
basep /= Dot11Elt(ID=0, len=len(ssid), info=ssid)
basp /\nwhile 1:
    tmpp = basep
    tmpp /= fuzz(Dot11Elt(ID=1))             # Send a packet every 1/10th of a second, 20 times
    sendp(p, count=20, inter=.1)
Metasploit 3.1 Framework Fuzzer

- Exploit framework written in Ruby
- Includes over 250 exploits, 118 payloads and auxiliary utilities
- Designed for Linux or Windows systems
- Integrates exploits with payloads for various compromise methods
  - Adduser payload: Creates a new administrative user
  - VNC Inject payload: Starts a VNC process on target
  - Metatpreter: Enhanced remote shell access on target
- Auxiliary utilities include fuzzing tools
Metasploit Probe Response Fuzzing

```plaintext
--[ msf v3.2-release
+ -- ---=[ 269 exploits - 118 payloads
+ -- ---=[ 17 encoders - 6 nops
= [ 48 aux

msf > use auxiliary/dos/wireless/fuzz_proberesp
msf auxiliary(fuzz_proberesp) > set ADDR_DST 00:13:ce:55:98:ef
ADDR_DST => 00:13:ce:55:98:ef
msf auxiliary(fuzz_proberesp) > set PING_HOST 10.0.0.2
PING_HOST => 10.0.0.2
msf auxiliary(fuzz_proberesp) > exploit
[*] Sending corrupt frames...
```
Metasploit Probe Response Fuzzing GUI

**CHANNEL**
The default channel number (type: integer)

**DRIVER**
The name of the wireless driver for lorcon (type: string)

**INTERFACE**
The name of the wireless interface (type: string)

**PING_HOST**
Ping the wired address of the target host (type: string)

Launch Auxiliary

ADVANCED OPTIONS
Driver Disassembly for Bug Hunting

```
loc_1D314:
    mov    al, [eax+edi+0FCBCh]
    mov    [ebx+45h], al
    movzx   ax, byte ptr [esi+9]
    movzx   cx, byte ptr [esi+8]
    push    0
    push    [ebp+var_C]
    push    [ebp+var_10]
    shl    eax, 8
    add    eax, ecx
    mov    [ebx+2Ah], ax
    call   sub_318D0
    test   eax, eax
    jz     loc_1D2AE
    mov    cl, [eax+1] ; SSID IE offset + 1 = length byte?
    mov    [ebx+6], cl
    movzx   ecx, cl ; Length of data to copy
    lea    esi, [eax+2] ; Data to be copied
    mov    eax, ecx
    shr    ecx, 2 ; Save the length for later
divide ecx by 4, DWORD size
    lea    edi, [ebx+7] ; Destination location on the stack
    lea    edi, [ebx+7]
    lea    edi, [ebx+7]
    lea    edi, [ebx+7]
    lea    edi, [ebx+7]
    rep movsd
    mov    ecx, eax
    rep movsb
    mov    esi, [ebp+var_C]
```
Exploiting Driver Bugs

- IEEE 802.11 fuzzing has uncovered driver bugs, attacker opportunities
- Drivers run in ring0, compromise reveals full access to host by the attacker
- Driver vulnerabilities are often not mitigated with encryption or authentication
  - Applicable regardless of WPA, WPA2, EAP/TLS, etc.
- Few organizations upgrade drivers as part of a patch management process
  - Systems remain vulnerable for an extended duration
Metasploit - Exploiting Driver Flaws

**Broadcom Wireless Driver Probe Response SSID Overflow**

This module exploits a stack overflow in the Broadcom Wireless driver that allows remote code execution in kernel mode by sending a 802.11 probe response that contains a long SSID. The target MAC address must be provided to use this exploit. The two cards tested fell into the 00:14:a5:06:XX:XX and 00:14:a4:2a:XX:XX ranges. This module depends on the Lorcon library and only works on the Linux platform with a supported wireless card. Please see the Ruby Lorcon documentation (external/ruby-lorcon/README) for more information.
Drivers and 802.11n Networks

• New complexities in 802.11n require new drivers to be written
  • New frame types, information elements, frame aggregation mechanisms, QoS parameters, etc.
  • Complexity is an attacker's friend

• Manufacturers in a frenzy of delivering 802.11n
  • Often, security wanes when product deadlines approach

802.11n represents new opportunities to exploit implementation flaws in drivers
802.11n Aggregate MSDU Delivery

- One of two mechanisms for aggregating traffic
- Multiple frames for any destination are aggregated into a single payload
  - AP or STA de-aggregates packets and processes data

<table>
<thead>
<tr>
<th>802.11 + QoS</th>
<th>A-MSDU</th>
<th>A-MSDU</th>
<th>A-MSDU</th>
</tr>
</thead>
<tbody>
<tr>
<td>Destination Address</td>
<td>Source Address</td>
<td>MSDU Length</td>
<td></td>
</tr>
<tr>
<td>6 bytes</td>
<td>6 bytes</td>
<td>2 bytes</td>
<td></td>
</tr>
</tbody>
</table>
Potential A-MSDU Handling Example

```c
handle_amsdu(uint8_t *packet, int framelen) {
    int offset = 0;
    struct amsdu_header *amsduhdr;

    while (framelen != 0) {
        amsduhdr = (packet+offset);
        if (memcmp(amsduhdr->destaddr, MY_MAC, 6)) {
            process_amsdu(amsduhdr+AMSDUHDR_LEN);
        }
        framelen -= amsduhdr->length;
        offset += amsduhdr->length;
    }
}
```

Attacker controls "length" in A-MSDU field, can influence framelen to become negative
New Metasploit Fuzzer - A-MSDU

• New fuzzer adds ability to fuzz test A-MSDU payloads
• Sends one initial payload, with following random MSDU length and payload

```plaintext
= [ msf v3.2-release
+ -- ---[ 269 exploits - 118 payloads
+ -- ---[ 17 encoders - 6 nops
= [ 48 aux

msf > use auxiliary/dos/wireless/fuzz_amsdu
msf auxiliary(fuzz_amsdu) > set ADDR_DST 00:1d:7e:03:28:bb
ADDR_DST => 00:1d:7e:03:28:bb
msf auxiliary(fuzz_amsdu) > set ADDR_SRC 00:19:5b:4e:29:bl
ADDR_SRC => 00:19:5b:4e:29:bl
msf auxiliary(fuzz_amsdu) > set PING_HOST 10.0.0.2
PING_HOST => 10.0.0.2
msf auxiliary(fuzz_amsdu) > exploit
[*] Sending corrupt frames...```
Mitigating Driver Flaws

• Ensure vendors are maintaining driver versions and responding to vulnerability reports
  • Monitor vendor website for driver updates
• Monitoring public wireless vulnerability reports
• Perform your own fuzzing tests
  • Write your own tools, leverage existing free and commercial tools
• Auditing your environment for driver vulnerabilities
• WiFiDEnum - Wireless Driver Enumerator
WiFiDEnum

Freely available from labs.arubanetworks.com/wifidenum
802.11n Risk Mitigation

• Careful deployment planning required
  • Always leverage WPA/WPA2 with strong EAP types for protected authentication

• Discuss with vendor WIDS strategies for channel monitoring, GF detection/mitigation

• No protection against DoS vulnerabilities (including 802.11w and MFP)

• Carefully monitor workstations for driver threats

• Consider in-house and commercial testing
Summary

- 802.11n promises to significantly enhance WLAN
- New application and cost savings opportunities
- Consistency in performance and reliability a huge win for organizations
- Improved bandwidth rivals or exceeds many existing LAN deployments
- Not without risks that can expose organizations
- Careful planning, vendor communication required for successful deployments
Questions? Thank you!

• Your Speaker:

Joshua Wright
Senior Security Researcher
Aruba Networks
jwright@arubanetworks.com
Office/Mobile: 401-524-2911

Knowledge helps us all to defend our networks