Device Drivers:
Don’t build a house on a shaky foundation

johnny cache, researcher
david maynor, SecureWorks
Overview

- Problems
- Nifty Fingerprinting Stuff
- Finding and Exploiting Vulns
- Shellcode Design
- DEMOS!!!!!!
Problems?

- Speed to market is so important.
- Some things don’t get tested properly
- New hardware and committee designed protocols are especially susceptible.
Problems (cont…)

• Although what follows is mostly focused on 802.11a/b/g the lessons learned can be applied to lots of things
  – Bluetooth
  – New 802.11 specs
  – Wireless data (EDGE, EV-DO, HSDPA)
802.11

- Why is it so complicated
- Does it have to be
- Can we fix it?
- Consequence’s of complexity:
  - Fingerprinting 802.11 implementations
  - Exploiting device drivers
Why so complicated?

- "Fear leads to anger. Anger leads to hate. Hate leads to protocols designed by committee." --warlord (?)
Why so complicated

- Partly to ambitious, partly attempting to deal with legitimate problems.
- hidden nodes
- unreliable links
- other networks on same channel
Can we fix it

• Yes, all it costs is standards compliance.
• Ignore management frames
• Ignore (some?) control frames
• Remove extra’s (more on these later),
Why is this interesting?

• Complexity is a hacker’s best friend.
• If its not complex theres no room for bugs. No bugs means no fun.
• 802.11 is not lacking in complexity.
Ethernet

- 3 fields: src, dst, type.
802.11

- Version
- Type
- Subtype
- 8 flags.
- 1, 2, 3 or 4 addresses, variable positions
- Frag num
- Sequence num
Not done yet..

• Positive acknowledgement
• 11 management frames
• 6 control frames
• ..lots of subtypes for each.
• ..various encryption fields (IV, MIC/ICV, etc)
More features!

- Ad-Hoc
- Power savings
- 2 types of MAC (PCF vs DCF)
- .11e QoS
- Geo-locating proposed? WTH does ‘media access control’ have to do with geo-locating
What do you get when you remove the extras?

Nintendo DS

- No Wi-Fi certification
- Nowhere near 802.11 compliant
- Ignores de-auth/disassociates
- Possibly ignores control packets
- Works great!

(probably doesn’t roam very well)
Fingerprinting 802.11

- Why bother
  - Target exploits
  - WIDS can monitor users’ chipset, driver.
  - Possibly refine OS fingerprints
Fingerprinting 802.11

• Why is this cool
  – No other link layer protocol fingerprints that I know of

• Why is this possible?
  – Complexity of the protocol
How far down can you go?

- Chipset families
- Distinct drivers for chipsets
- Different versions of the same driver
- Firmware (?)
Specific fingerprints

- RTS/CTS window honouring
- Association Redirection
- Duration analysis
RTS/CTS

- RTS/CTS packets used to reserve media for large enough packets.
RTS/CTS

A

1 - A sends large frame

B
RTS/CTS

2 - halfway through, B transmits
RTS/CTS

4 - A sends RTS

"I need the air for 20000 usec"
RTS/CTS

A needs air for 20000 usec. Everyone shutup!

5 - AP sends CTS
RTS/CTS

6- A sends large frame, B stays quiet for 20000 usec
RTS/CTS

A finishes, B transmits when he's done
How many implementations use this?

- Most? Nope.
- A few? Nope
- None? Yes!

(under normal conditions)
RTS/CTS

• If they didn’t bother to implement it, they care if other people have?
RTS/CTS

• Though code was written to analyze packet dumps, results were not deterministic enough to be useful.
• Getting such a high resolution clock/timestamp very difficult.
Association Redirection

- Active fingerprinting technique.
- High resolution.
- Mind-numbingly boring to automate.
Association Redirection

- Specified in standard: pg 376
Quick Overview

Important 802.11 fields:
Src, Dst, BSSID
Typical 802.11 Traffic

10.0.0.100
00:11:95:C2:E7:8A
10.0.0.222
00:10:C6:6B:07:1D

10.0.0.1
BSSID: 00:30:BD:C0:38:9A

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>23</td>
<td>0.25343</td>
<td>10.0.0.100</td>
<td>10.0.0.222</td>
<td>TCP</td>
<td>50300 &gt; 50300</td>
</tr>
<tr>
<td>24</td>
<td>0.25476</td>
<td>10.0.0.100</td>
<td>10.0.0.222</td>
<td>TCP</td>
<td>50300 &gt; 50300</td>
</tr>
</tbody>
</table>

Frame 23 (80 bytes on wire, 80 bytes captured)

IEEE 802.11
  Type/Subtype: Data (32)
  Frame Control: 0x0108 (Normal)
  Duration: 258
  BSSID: 00:30:bd:c0:38:9a (BelkinCo_c0:38:9a)
  Source address: 00:11:95:c2:e7:8a (AlphaNet_c2:e7:8a)
  Destination address: 00:10:c6:6b:07:1d (Usi_6b:07:1d)
  Fragment number: 0
  Sequence number: 3368

Logical-Link Control

Internet Protocol, Src Addr: 10.0.0.100 (10.0.0.100), Dst Addr: 10.0.0.222 (10.0.0.222)
Normal 802.11 Association
Association Redirection

Successful

Unsuccessful

Client
- Auth Req
  DST 00:11:22:33:44:55
- Auth Reply
  SRC 00:11:22:33:44:55
- Assoc Req
  DST 00:11:22:33:44:55
- Assoc Reply
  SRC = 00:22:22:22:22
- DATA
  BSSID = 00:22:22:22:22

AP

Client
- Auth Req
  DST 00:11:22:33:44:55
- Auth Reply
- Assoc Req
  DST 00:11:22:33:44:55
- Assoc Reply
  SRC = 00:22:22:22:22
- DATA
  BSSID = 00:11:22:33:44:55

AP
<table>
<thead>
<tr>
<th>id-num</th>
<th>image</th>
<th>MAC/Model/Chipset</th>
<th>driver-id</th>
<th>SRC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><img src="image1.png" alt="Image" /></td>
<td>00:12:17:79:1C:B0 Atheros AR5212</td>
<td>ar5211.sys</td>
<td>IGN_ASSOC_REPLY</td>
</tr>
<tr>
<td>2</td>
<td><img src="image2.png" alt="Image" /></td>
<td>00:20:A6:4C:D9:4A Atheros AR5212</td>
<td>ntp11ag.sys</td>
<td>IGN_ASSOC_REPLY</td>
</tr>
<tr>
<td>3</td>
<td><img src="image3.png" alt="Image" /></td>
<td>00:20:A6:4B:DD:85 Atheros AR5211</td>
<td>(ntpr11ag.sys)</td>
<td>IGN_ASSOC_REPLY</td>
</tr>
<tr>
<td>4</td>
<td><img src="image4.png" alt="Image" /></td>
<td>00:20:A6:51:EC:09 Atheros AR5212</td>
<td>(ntpr11ag.sys)</td>
<td>IGN_ASSOC_REPLY</td>
</tr>
<tr>
<td>5</td>
<td><img src="image5.png" alt="Image" /></td>
<td>00:0A:95:F3:2F:AB Broadcom BCM4318</td>
<td>AppleAirport2-bcm4318</td>
<td>DEAUTH_FLOOD_NULL</td>
</tr>
<tr>
<td>6</td>
<td><img src="image6.png" alt="Image" /></td>
<td>00:14:a5:06:8F:E6 BCM-4306</td>
<td>BCMWL5.sys</td>
<td>DEAUTH_FLOOD_NULL</td>
</tr>
<tr>
<td>7</td>
<td><img src="image7.png" alt="Image" /></td>
<td>00:0E:35:E9:C9:5B Intel PRO/Wireless 2200BG</td>
<td>w29n51.sys</td>
<td>DUAL_NACK_DATA</td>
</tr>
<tr>
<td>8</td>
<td><img src="image8.png" alt="Image" /></td>
<td>00:13:46:E3:B4:2C Ralink RA2570</td>
<td>rt2500usb.sys</td>
<td>IGN_ASSOC_REPLY</td>
</tr>
<tr>
<td>9</td>
<td><img src="image9.png" alt="Image" /></td>
<td>00:04:E2:80:2C:21 Prism 2.5</td>
<td>smc2532w.sys</td>
<td>DEAUTH_TYPE_1</td>
</tr>
<tr>
<td>10</td>
<td><img src="image10.png" alt="Image" /></td>
<td>00:14:A4:2A:9E:58 BCM4318</td>
<td>bcmwl5.sys</td>
<td>DEAUTH_FLOOD_NULL</td>
</tr>
</tbody>
</table>
So what weird things happen?

- Cards de-auth flood null address (broadcom)
- Cards think they are on both networks? (centrino)
- Other less dramatic hijinks.
Deauth-Flood example
auth-reply

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>1.315883</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 8i</td>
<td>Authentication</td>
</tr>
<tr>
<td>41</td>
<td>1.316220</td>
<td>AppleCom_f3:2f:ab (R)</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 8i</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>42</td>
<td>1.317122</td>
<td>Cimsys_33:44:55</td>
<td>AppleCom_f3:2f:ab</td>
<td>IEEE 8i</td>
<td>Authentication</td>
</tr>
<tr>
<td>44</td>
<td>1.318342</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 8i</td>
<td>Association Request, SSID: &quot;dojooffoo&quot;</td>
</tr>
<tr>
<td>45</td>
<td>1.318679</td>
<td>AppleCom_f3:2f:ab (R)</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 8i</td>
<td>Acknowledgement</td>
</tr>
</tbody>
</table>

Frame 42 (30 bytes on wire, 30 bytes captured)

- IEEE 802.11
- Type/Subtype: Authentication (11)
- Frame Control: 0x00B0 (Normal)
- Duration: 314
- Destination address: 00:0a:95:f3:2f:ab (AppleCom_f3:2f:ab)
- Source address: 00:11:22:33:44:55 (Cimsys_33:44:55)
- BSS Id: 00:11:22:33:44:55 (Cimsys_33:44:55)
- Fragment number: 0
- Sequence number: 108
Deauth-Flood example

assoc-request

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>1.315883</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 8i</td>
<td>Authentication</td>
</tr>
<tr>
<td>41</td>
<td>1.316220</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 8i</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>42</td>
<td>1.317122</td>
<td>Cimsys_33:44:55</td>
<td>AppleCom_f3:2f:ab</td>
<td>IEEE 8i</td>
<td>Authentication</td>
</tr>
<tr>
<td>43</td>
<td>1.317466</td>
<td>Cimsys_33:44:55</td>
<td>AppleCom_f3:2f:ab</td>
<td>IEEE 8i</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>44</td>
<td>1.318342</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 8i Association Request, SSID: &quot;dojooffoo&quot;</td>
<td></td>
</tr>
<tr>
<td>45</td>
<td>1.318679</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 8i</td>
<td>Acknowledgement</td>
</tr>
</tbody>
</table>

IEEE 802.11

Type/Subtype: Association Request (0)
Frame Control: 0x0000 (Normal)
Duration: 314
Destination address: 00:11:22:33:44:55 (Cimsys_33:44:55)
Source address: 00:0a:95:f3:2f:ab (AppleCom_f3:2f:ab)
BSS Id: 00:11:22:33:44:55 (Cimsys_33:44:55)
Fragment number: 0
Sequence number: 46
IEEE 802.11 wireless LAN management frame
Deauth-Flood example

assoc-reply

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>1.315883</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 802.11</td>
<td>Authentication</td>
</tr>
<tr>
<td>41</td>
<td>1.316220</td>
<td>AppleCom_f3:2f:ab (R)</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 802.11</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>42</td>
<td>1.317122</td>
<td>Cimsys_33:44:55</td>
<td>AppleCom_f3:2f:ab</td>
<td>IEEE 802.11</td>
<td>Authentication</td>
</tr>
<tr>
<td>43</td>
<td>1.317466</td>
<td>Cimsys_33:44:55 (RA)</td>
<td>AppleCom_f3:2f:ab (R)</td>
<td>IEEE 802.11</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>44</td>
<td>1.318342</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 802.11</td>
<td>Association Request, SSID: &quot;dojoofoo&quot;</td>
</tr>
<tr>
<td>45</td>
<td>1.318679</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 802.11</td>
<td>Acknowledgement</td>
</tr>
</tbody>
</table>

IEEE 802.11

Type/Subtype: Association Response (1)

Frame Control: 0x0010 (Normal)
Duration: 258
Destination address: 00:0a:95:f3:2f:ab (AppleCom_f3:2f:ab)

BSS Id: 00:11:22:33:44:55 (Cimsys_33:44:55)
Fragment number: 0
Sequence number: 109

IEEE 802.11 wireless LAN management frame
Deuath-Flood starts

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>Source</th>
<th>Destination</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>40</td>
<td>1.315883</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 80.11</td>
<td>Authentication</td>
</tr>
<tr>
<td>41</td>
<td>1.316220</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 80.11</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>42</td>
<td>1.317122</td>
<td>Cimsys_33:44:55</td>
<td>AppleCom_f3:2f:ab</td>
<td>IEEE 80.11</td>
<td>Authentication</td>
</tr>
<tr>
<td>43</td>
<td>1.317466</td>
<td>Cimsys_33:44:55</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 80.11</td>
<td>Acknowledgement</td>
</tr>
<tr>
<td>44</td>
<td>1.318342</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 80.11</td>
<td>Association Request, SSID: &quot;dojooffoo&quot;</td>
</tr>
<tr>
<td>45</td>
<td>1.318679</td>
<td>AppleCom_f3:2f:ab</td>
<td>Cimsys_33:44:55</td>
<td>IEEE 80.11</td>
<td>Acknowledgement</td>
</tr>
</tbody>
</table>

IEEE 802.11
Type/Subtype: Deauthentication (12)
- Frame Control: 0x00C0 (Normal)
  - Duration: 314
  - Source address: 00:0a:95:f3:2f:ab (AppleCom_f3:2f:ab)
  - BSSID: 00:00:00:00:00:00 (00:00:00:00:00:00)
  - Fragment number: 0
  - Sequence number: 47
- IEEE 802.11 wireless LAN management frame
Association Redirection redux

• If 1 weird standards quirk is good 3 must be better!
  – Instead of just source mangle as many things as possible: src, bssid, both
Table 2 here
Association Redir redux

• If 3 standards quirks work OK, why not 9?
• Two more tables
Tables 3 and 4 here
Association Redirection summary

- very possible to remotely version chipset
- can’t really distinguish different drivers
- active technique, requires you to transmit packets.
Duration analysis

- Totally passive
- Very accurate
- Easy to automate
- Only basic statistical techniques used.
What is a duration?

<table>
<thead>
<tr>
<th>No.</th>
<th>Time</th>
<th>HW-src</th>
<th>HW-dst</th>
<th>Protocol</th>
<th>Info</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>21:07:18.620</td>
<td>00:0a:95:f3:2f:ab</td>
<td>ff:ff:ff:ff:ff:ff</td>
<td>IEEE 8 Data</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>21:07:21.388</td>
<td>00:0a:95:f3:2f:ab</td>
<td>ff:ff:ff:ff:ff:ff</td>
<td>IEEE 8 Data</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>21:07:23.428</td>
<td>00:0a:95:f3:2f:ab</td>
<td>ff:ff:ff:ff:ff:ff</td>
<td>IEEE 8 Data</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>21:07:23.429</td>
<td>00:0a:95:f3:2f:ab</td>
<td>ff:ff:ff:ff:ff:ff</td>
<td>IEEE 8 Data</td>
<td></td>
</tr>
</tbody>
</table>

Frame 3 (68 bytes on wire, 68 bytes captured)

IEEE 802.11
- Type/Subtype: Data (32)
- Frame Control: 0x4108 (Normal)
- Duration: 258

BSSID: 00:30:bd:c0:38:9a (00:30:bd:c0:38:9a)
Source address: 00:0a:95:f3:2f:ab (00:0a:95:f3:2f:ab)
Destination address: ff:ff:ff:ff:ff:ff (ff:ff:ff:ff:ff:ff)
Fragment number: 0
Sequence number: 1286

WFP parameters
What influences duration values.

- Rate (.11b, .11g)
- Short slot time (g only)
- Short pre amble
Example atheros fingerprint

Well behaved atheros card:

CTS: 0
pwrmgmt: 1
frag: 0
order: 0

---------

<0 0>  Duration( (314) )  //assoc request
<0 4>  Duration( (0) (314) )  //probe request
<0 11> Duration( (314) )  //authentication
<2 0>  Duration( (162) (0) )  //data
<2 4>  Duration( (162) )  //null function data
Example prism fingerprint

poorly behaved prism card:

- CTS: 0
- pwrmgmt: 1
- frag: 0
- order: 0

--------

<0 0>  Duration( (258) )  //assoc req
<0 4>  Duration( (0) )    //probe req
<0 11> Duration( (53389) )  //auth
<0 12> Duration( (258) (314) )  //de-auth
<2 0>  Duration( (213) (0) (223) )  //data
<2 4>  Duration( (37554) )    //null-func
Simple example

- Duration match 2 prints here

<table>
<thead>
<tr>
<th>Score:</th>
<th>ID</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>29800.00</td>
<td>1</td>
<td>atheros card</td>
</tr>
<tr>
<td>13000.00</td>
<td>9</td>
<td>prism card</td>
</tr>
</tbody>
</table>

Score: ID
50600.00 9 //prism card
13000.00 1 //atheros card
Simple example cont.

```
Score: ID    Score: ID
24802.86 2 //Atheros
23785.00 1 //Atheros
23785.00 3 //Atheros
23785.00 4 //Atheros
19645.71 7 //centrino
17610.00 10 //broadcom
16110.00 6 //broadcom
13327.86 13 //aironet 350
12880.00 12 //Ninteno DS
12310.00 5 //broadcom
10077.86 9 //prism
9060.00 8 //ralink
8577.86 11 //PSP
```

```
Score: ID
172256.43 9 //prism
21138.57 10 //broadcom
18982.62 11 //PSP
17738.57 5 //broadcom
11774.29 2 //atheros
10417.14 7 //centrino
10161.19 13 //aironet 350
10138.57 6 //broadcom
10077.86 4 //atheros
10077.86 3 //atheros
10077.86 1 //atheros
9060.00 8 //ralink
5280.00 12 //Nintendo DS
```
### Real life example (centrino)

```bash
../../duration-print-matcher -a 00:0E:35:E9:C9:5B  -p
./pcaps/7-2-mixed--wrt54g.pcap -f ./prints/*.print
```

<table>
<thead>
<tr>
<th>Score</th>
<th>ID</th>
<th>Device</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>58404.88</strong></td>
<td>7</td>
<td><code>//centrino</code></td>
</tr>
<tr>
<td>44365.05</td>
<td>13</td>
<td><code>//aironet 350</code></td>
</tr>
<tr>
<td>38781.71</td>
<td>9</td>
<td><code>//prism2.5</code></td>
</tr>
<tr>
<td>37394.32</td>
<td>6</td>
<td><code>//broadcom 4306</code></td>
</tr>
<tr>
<td>32883.61</td>
<td>4</td>
<td><code>//atheros ar5212</code></td>
</tr>
<tr>
<td>32883.61</td>
<td>2</td>
<td><code>//atheros ar5212</code></td>
</tr>
<tr>
<td>31553.30</td>
<td>5</td>
<td><code>//broadcom (apple)</code></td>
</tr>
<tr>
<td>26508.61</td>
<td>1</td>
<td><code>//atheros 5212</code></td>
</tr>
<tr>
<td>24100.27</td>
<td>10</td>
<td><code>//broadcom bcm4318</code></td>
</tr>
<tr>
<td>22051.56</td>
<td>3</td>
<td><code>//atheros ar5212</code></td>
</tr>
<tr>
<td>19526.16</td>
<td>11</td>
<td><code>//PSP</code></td>
</tr>
<tr>
<td>14103.69</td>
<td>8</td>
<td><code>//Ralink RA2570</code></td>
</tr>
<tr>
<td>12248.00</td>
<td>12</td>
<td><code>//Nintendo DS</code></td>
</tr>
</tbody>
</table>
Unknown Ralink example

tcpcap -i rausb0 -s 0 -w unknown.pcap

```
../duration-print-matcher -a 00:13:D4:6F:13:A9 -p ./unknown.pcap -f ./prints/*.prnt
```

<table>
<thead>
<tr>
<th>Score</th>
<th>ID</th>
<th>Manufacturer</th>
</tr>
</thead>
<tbody>
<tr>
<td>84719.08</td>
<td>8</td>
<td>Ralink</td>
</tr>
<tr>
<td>14103.69</td>
<td>4</td>
<td>atheros</td>
</tr>
<tr>
<td>14103.69</td>
<td>2</td>
<td>atheros</td>
</tr>
<tr>
<td>14103.69</td>
<td>10</td>
<td>broadcom</td>
</tr>
<tr>
<td>14103.69</td>
<td>1</td>
<td>atheros</td>
</tr>
<tr>
<td>13319.08</td>
<td>7</td>
<td>centrino</td>
</tr>
<tr>
<td>12603.69</td>
<td>6</td>
<td>broadcom</td>
</tr>
<tr>
<td>11819.08</td>
<td>5</td>
<td>broadcom</td>
</tr>
<tr>
<td>9396.00</td>
<td>9</td>
<td>prism</td>
</tr>
<tr>
<td>9396.00</td>
<td>3</td>
<td>atheros</td>
</tr>
<tr>
<td>7896.00</td>
<td>13</td>
<td>aironet</td>
</tr>
<tr>
<td>7896.00</td>
<td>11</td>
<td>PSP</td>
</tr>
<tr>
<td>5448.00</td>
<td>12</td>
<td>Nintendo DS</td>
</tr>
</tbody>
</table>
So how’s it work?

--MagicStats Duration summary---
Total number of unique durations: 12
Total volume: 95

<table>
<thead>
<tr>
<th>dur</th>
<th>times_seen</th>
<th>prob</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>0,</td>
<td>25,</td>
<td>0.2632,</td>
<td>3.8000</td>
</tr>
<tr>
<td>117,</td>
<td>8,</td>
<td>0.0842,</td>
<td>11.8750</td>
</tr>
<tr>
<td>127,</td>
<td>2,</td>
<td>0.0211,</td>
<td>47.5000</td>
</tr>
<tr>
<td>152,</td>
<td>1,</td>
<td>0.0105,</td>
<td>95.0000</td>
</tr>
<tr>
<td>162,</td>
<td>15,</td>
<td>0.1579,</td>
<td>6.3333</td>
</tr>
<tr>
<td>213,</td>
<td>5,</td>
<td>0.0526,</td>
<td>19.0000</td>
</tr>
<tr>
<td>223,</td>
<td>1,</td>
<td>0.0105,</td>
<td>95.0000</td>
</tr>
<tr>
<td>248,</td>
<td>2,</td>
<td>0.0211,</td>
<td>47.5000</td>
</tr>
<tr>
<td>258,</td>
<td>6,</td>
<td>0.0632,</td>
<td>15.8333</td>
</tr>
<tr>
<td>314,</td>
<td>28,</td>
<td>0.2947,</td>
<td>3.3929</td>
</tr>
<tr>
<td>37554,</td>
<td>1,</td>
<td>0.0105,</td>
<td>95.0000</td>
</tr>
<tr>
<td>53389,</td>
<td>1,</td>
<td>0.0105,</td>
<td>95.0000</td>
</tr>
</tbody>
</table>

Atheros print

CTS: 0
pwrmgmt: 1
frag: 0
order: 0

<0 0> Duration( (314) )
<0 4> Duration( (0) (314) )
<0 11> Duration( (314) )

<2 0> Duration( (162) (0) )
<2 4> Duration( (162) )
So how’s it work?

• Compute fingerprint across input pcap.
• Fuzzilly compare it to all known fingerprints.
  – For every matching duration in comparison print, add points proportional to weight for that duration.
  – Bonus points for matching type, subtype, and duration all at once.
Fuzzy compare

- For every matching duration in comparison print, add points proportional to weight for that duration.
- Bonus points for matching type, subtype, and duration all at once.
Also tracks a few other flags

<table>
<thead>
<tr>
<th>Flag</th>
<th>value</th>
<th>ratio</th>
<th>prob</th>
<th>weight</th>
</tr>
</thead>
<tbody>
<tr>
<td>CTS:</td>
<td>1</td>
<td>0/12</td>
<td>0.0000</td>
<td>inf</td>
</tr>
<tr>
<td>CTS:</td>
<td>0</td>
<td>12/12</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>PwrMgmt:</td>
<td>1</td>
<td>8/12</td>
<td>0.6667</td>
<td>1.5000</td>
</tr>
<tr>
<td>PwrMgmt:</td>
<td>0</td>
<td>4/12</td>
<td>0.3333</td>
<td>3.0000</td>
</tr>
<tr>
<td>frag:</td>
<td>1</td>
<td>0/12</td>
<td>0.0000</td>
<td>inf</td>
</tr>
<tr>
<td>frag:</td>
<td>0</td>
<td>12/12</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
<tr>
<td>order:</td>
<td>1</td>
<td>0/12</td>
<td>0.0000</td>
<td>inf</td>
</tr>
<tr>
<td>order:</td>
<td>0</td>
<td>12/12</td>
<td>1.0000</td>
<td>1.0000</td>
</tr>
</tbody>
</table>
how accurate is it?

- When run across my own set of training data, the following results apply:
  - **B-only (0x0021 flags, lexie)**
    - 26 times better than random
  - **mixed-BG (0x0401/0x0001 flags)**
    - 18 times better than random
Finding and exploiting vulns in drivers.
Ways to find bugs?

- Static auditing
- Fuzzing
Things to think about

- Fuzzing can be frustrating
  - A bug could be triggered by something 8 packet chains ago
  - Hard to track down in ring0
fuzz-e

( johnycsh@diz:fuzz-e )$./fuzz-e -R -A -P ath0 -n 500 -r rt2570 -i rausb0 -c 11 -D ./dest-addys.txt -w u20000 -s 00:07:0E:B9:74:BB -b 00:07:0E:B9:74:BB -E log.txt

-R random delays
-A autonomous mode (don’t stop)
-P passive interface to sniff on
-n 500 send 500 packets per cycle
-r rt2570 driver to inject with
-i rausb0 inject on rausb0
-c 11 set channel to 11
-D dest-addys specify list of victims
-w u20000 wait 200000 usecs (max)
-s source address of packets
-b bssid of packets
-E log events to log.txt
Shellcode

- Most often a direct return shell is not possible.
- Bots or other malicious shellcode have to be designed.
DEMOS

(there are a few)