802.11: Use, Misuse and the Need for a Robust Security Toolkit

David Pollino

Mike Schiffman

May 2002

@stake

Where Security & Business Intersect℠

Agenda

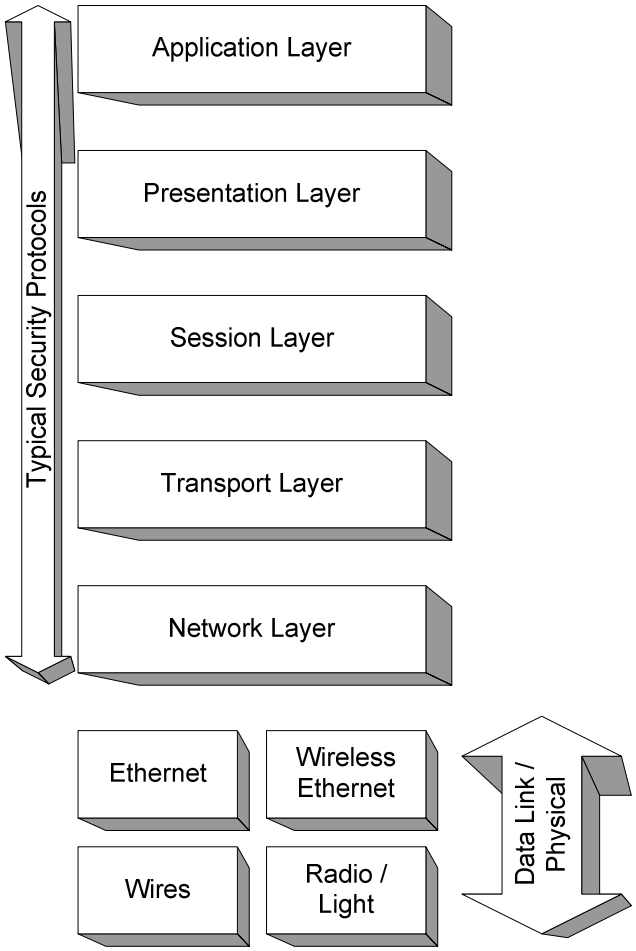
- Introductions
- WLAN Protocol Overview
- Current Latent Issues
- Protocol Flaws
- Existing Tools
- Gap analysis – The need for custom auditing tools
- Radiate
- Questions
- See you at the bar

Introductions

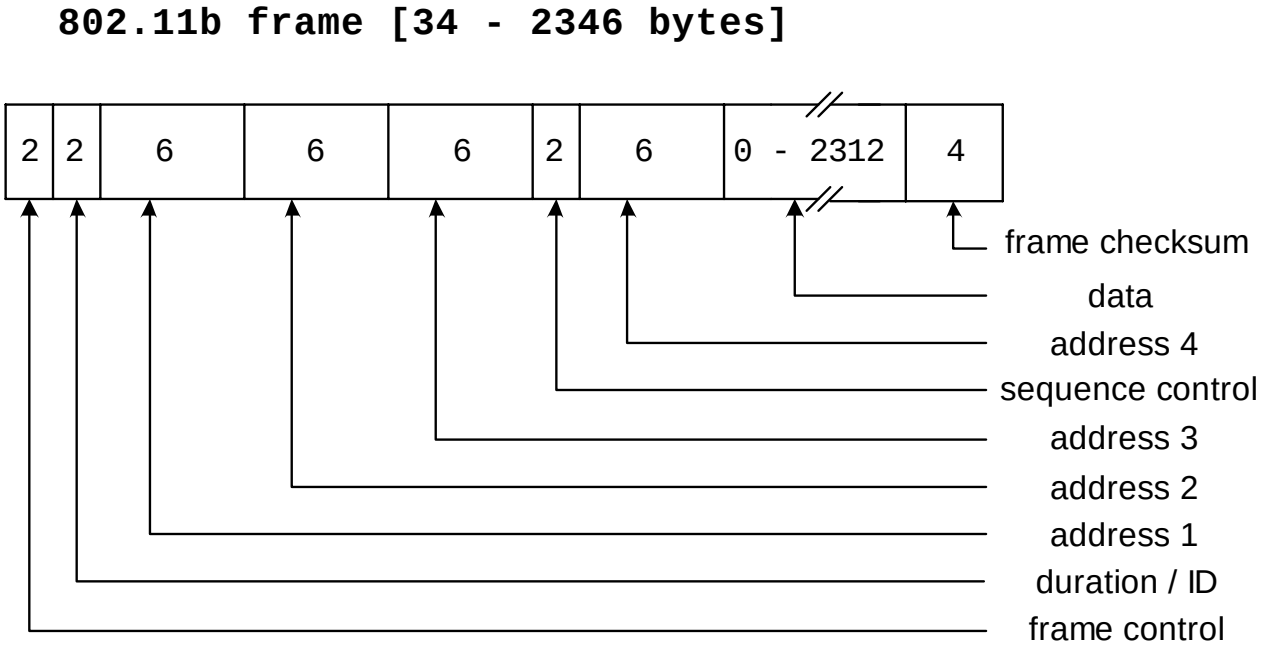
- **David Pollino**
  - dpollino@atstake.com
- **Mike Schiffman**
  - mike@atstake.com

WLAN protocol overview

- **Assumption: You know Ethernet**
- **Definition: Wireless local area network**
- **IEEE 802.11**
  - **Originally published in 1997**
  - **Ratified 1999**
  - **Security Mechanisms defined**
- **Widely used security protocols exist layer 3 and up**
  - IPSec, TLS, SSL, SSH
- **Few security protocols with limited application on layer 2**
  - PAP / CHAP / EAP for PPP



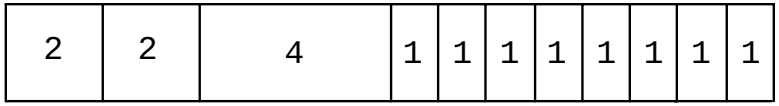
Different from Ethernet



- **Carrier Sense Multiple Access / Collision Detect with Ethernet**
- **Carrier Sense Multiple Access / Collision Avoidance with WLAN**
- **More addressing needed**
  - Wireless sender and receiver, not always the MAC sender and receiver

802.11b Types and Subtypes

802.11b frame control types and subtypes




```
   00 - management                    01 - control
0000 - association request       1011 - RTS (request to send)
0001 - association response      1100 - CTS (clear to send)
0010 - reassociation request     1101 - acknowledgement
0011 - reassociation response
0100 - probe request
0101 - probe response
1000 - beacon                      10 - data
1010 - disassociation            0000 - data
1011 - authentication
1100 - deauthentication
```

- **Roaming / Hidden stations different from Ethernet**
- **Reliability mechanisms on WLAN**

Standard 802.11 – Access Point Discovery

Beacon

Probe Request

Probe Response

Authentication Request

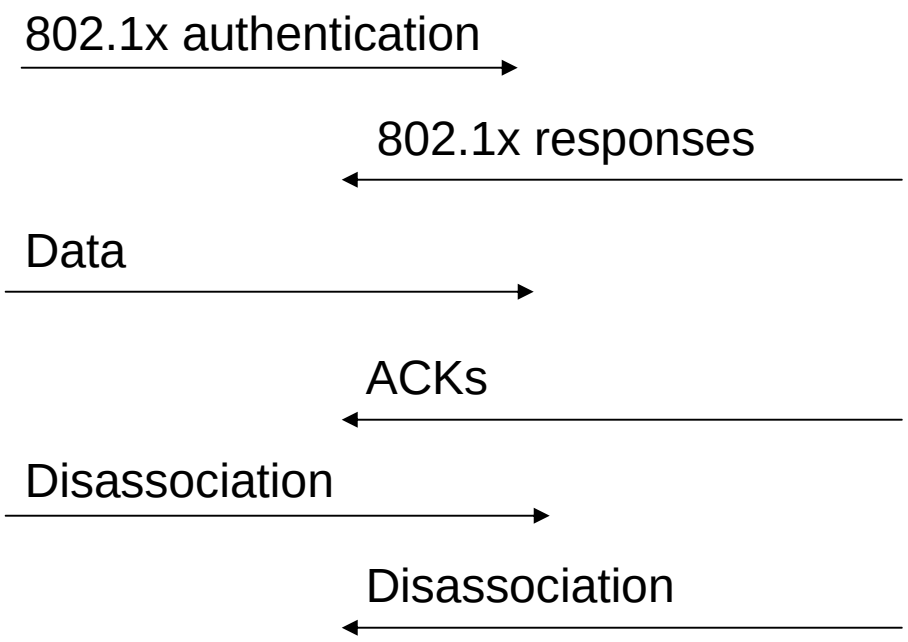
Authentication Response

Association Request

Association Response

- **Beacons sent out 10x second**
  - Advertise capabilities
- **Station queries access points**
  - Requests features
- **Access points respond**
  - With supported features
- **Authentication just a formality**
  - May involve more frames
- **Features used by war driving software**

Standard 802.11 – Session Continued



- **802.1x authentication**
  - More frames involved
  - May have multiple levels
- **802.1x can be used for WEP**
  - Dynamic key
- **Frames are ACK'd**
  - Not shown until now
- **Disassociations**
  - Used for roaming

Massive information leakage

Beacons

Probe Request

Probe Response

Association Request

Association Response

802.1x Request

802.1x Response

- **Beacons**
  - SSID, AP name, WEP used
  - Proprietary information
- **Probe Request**
  - Advertise existence of “hidden” network
  - Requests features, SSID
- **Association**
  - SSID passed in clear, even in closed systems
- **802.1x authentication**
  - Username may be in clear
  - Dictionary attacks against passwords?
  - Material for dynamic keys

Disassociation



Association

Data

ACK

Data

Spoofed Disassociation

Data

Data

Data

Session Hijacking

Review of Protocol Flaws

- **Trivial Denial of Service**
  - Session are torn down with a single frame
  - Many frames needed to regain session
- **Can force authentication**
  - Can be used to harvest usernames or password hashes
- **Can hijack sessions**
  - Unencrypted sessions vulnerable
- **Some 802.1x implementations vulnerable to dictionary attack**
  - Good passwords needed!
- **Can replay authentications for Denial of service**
  - May lead to denial of service from account lock out features
- **The Attacker controls network events = BAD**

Conclusion – Wireless protocol still flawed, but…

- **Risks can be minimized if best practices followed**
  - Use effective encryption – IPSec, 802.1x w/TKIP dynamic WEP
  - Authenticate user sessions
    - Good passwords – traditional dictionary and brute force attacks
  - Regularly assess network
    - Incorrect configuration
    - User installed access points
  - Keep meaningful logs
- **Progress is being made**
  - 802.1x, TKIP
  - Many Implementations still proprietary, some interoperability starting

WLANs – To buy or not buy?

- **Protocol insecurities need to be considered before deploying WLANs**
- **Secure WLANs are possible, there are a number of mechanisms available for security**
  - 802.1x with enhanced 802.11 features (TKIP, EAP/TLS)
  - IPSec VPNs, SSH, SSL
- **The cost of implementation is low and continues to drop**
  - Less that $250 for AP and card
- **Great benefits can be realized by implementing WLANs**
- **So, if you do not do it…**
- **Your users may to it for you**
  - Electronics store AP with no security enabled
  - Regular assessments needed to guard against insecure installations

@stake research

- **Wireless Center of Excellence**
  - WLAN
  - WAP, GPRS, UMTS
- **Buy my book**
- **Questions – time permitting**
- **Here's Mike!**

Overview

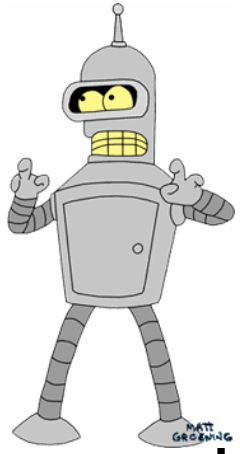
- **802.11-based networks are incredibly useful**
  - Many new products and services on top of them (cordless phones, PDAs)
  - WLANs can be quickly deployed
  - Mobility for users
  - Going to continue to grow and gain acceptance
    - Newer, faster physical interfaces being turned out
      - Same layer 2 protocols
- **There are obviously many security issues**
- **Sure, we need a way to be able to test for these issues**
  - Some tools currently exist
- **But what we really need a way to be able to test for arbitrary security issues**
- **We need a generic toolkit!**

Existing 802.11 Network Security Tools

- **AiroPeek / AiroPeek NX**
  - Wireless frame sniffer / analyzer
- **AirTraf**
  - Wireless sniffer / analyzer / “IDS”
- **AirSnort**
  - WEP key “cracker”
- **BSD Airtools**
  - Ports for common wireless tools, very useful
- **NetStumbler**
  - Access point enumeration tool

Where Existing Tools Come Up Lacking

- **Very task-oriented and specific**
- **Closed-source tools are not tunable**
  - "I'd like to use NetStumbler to listen for Beacons"
  - TOUGH LUCK DORKUS!

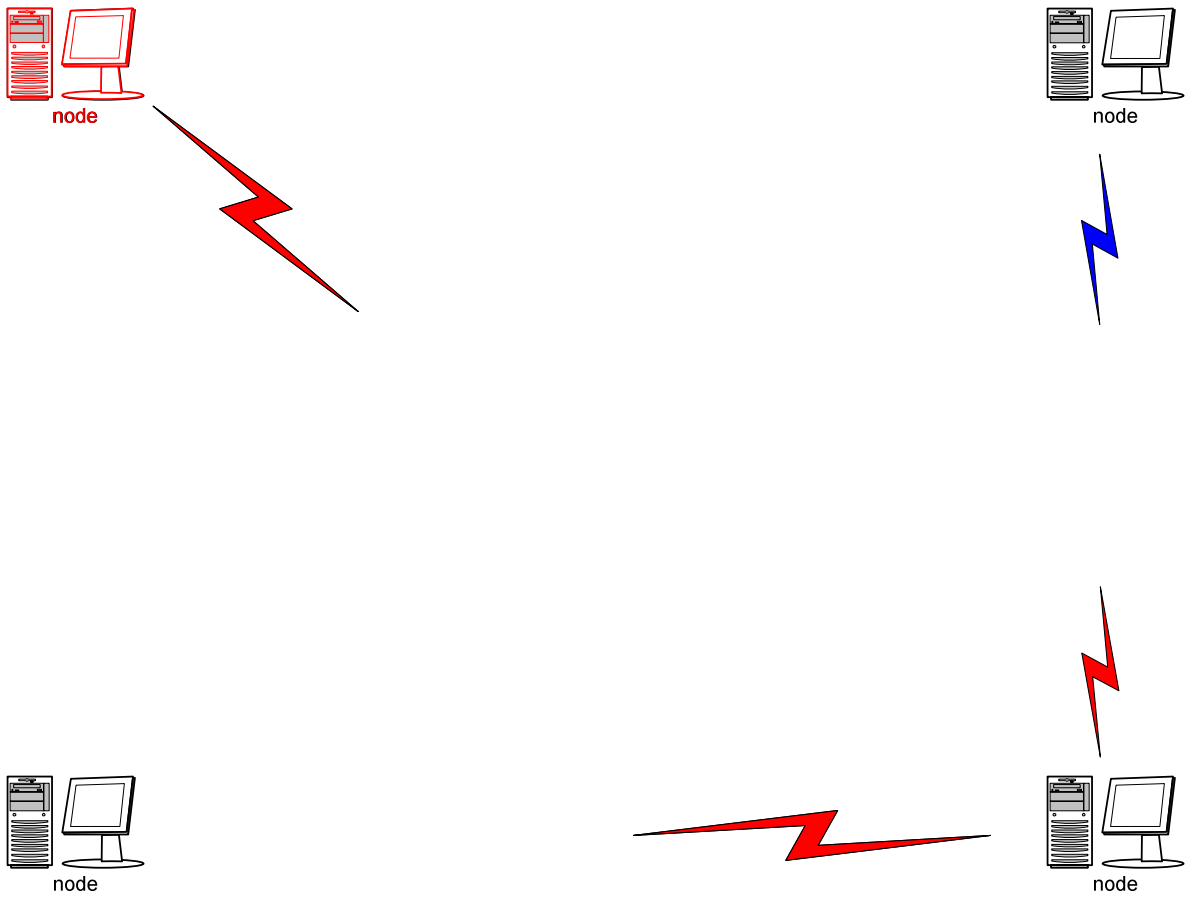


- **In order to test for unspecified security issues there needs to be a generic testing framework…**

Case Study: The Byzantine Generals Problem

- **It is important for an 802.11 network to be robust**
  - "Strongly formed or constructed"…
- **Byzantine Robustness means proof against Byzantine failure**
  - Byzantine generals problem (3t + 1)

Byzantine Failure



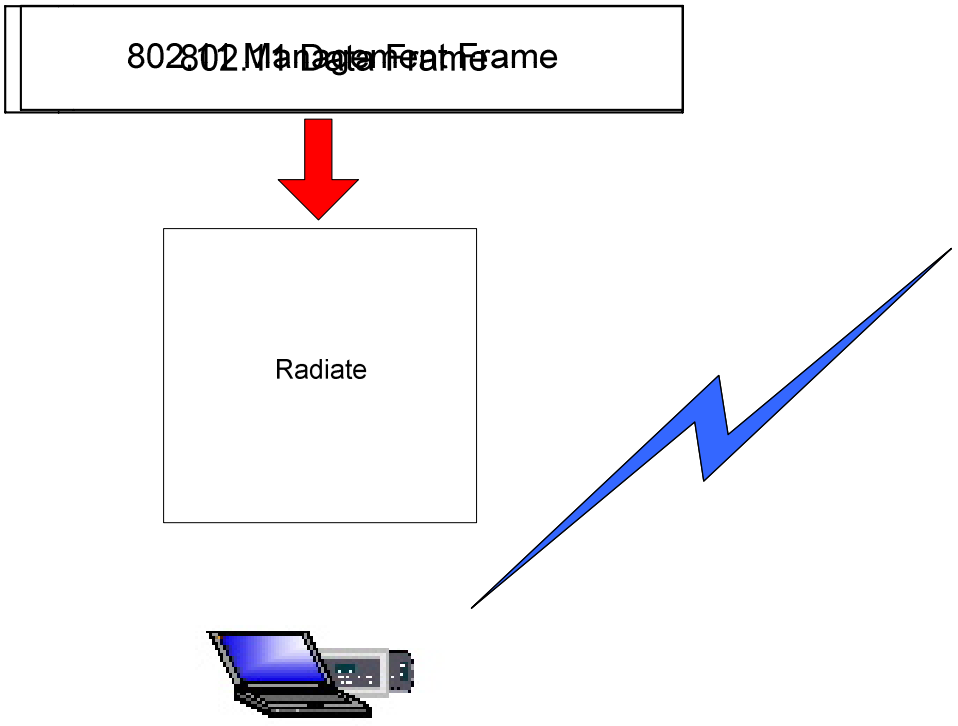
Byzantine Failure shows the need for a tool framework

- Byzantine fault injection
  - The Omerta Attack described earlier is an example of Byzantine failure (fault injection)
- In order to ferret out Byzantine failure situations we need arbitrary security tools
- A generic toolkit would allow an application developer to build tools to test for Byzantine failures via controlled Byzantine fault injection
- One example of such a toolkit for wired networks would be my handsome friend Libnet…
- HRM. WHAT COULD ALL OF THIS BE LEADING UP TO…?

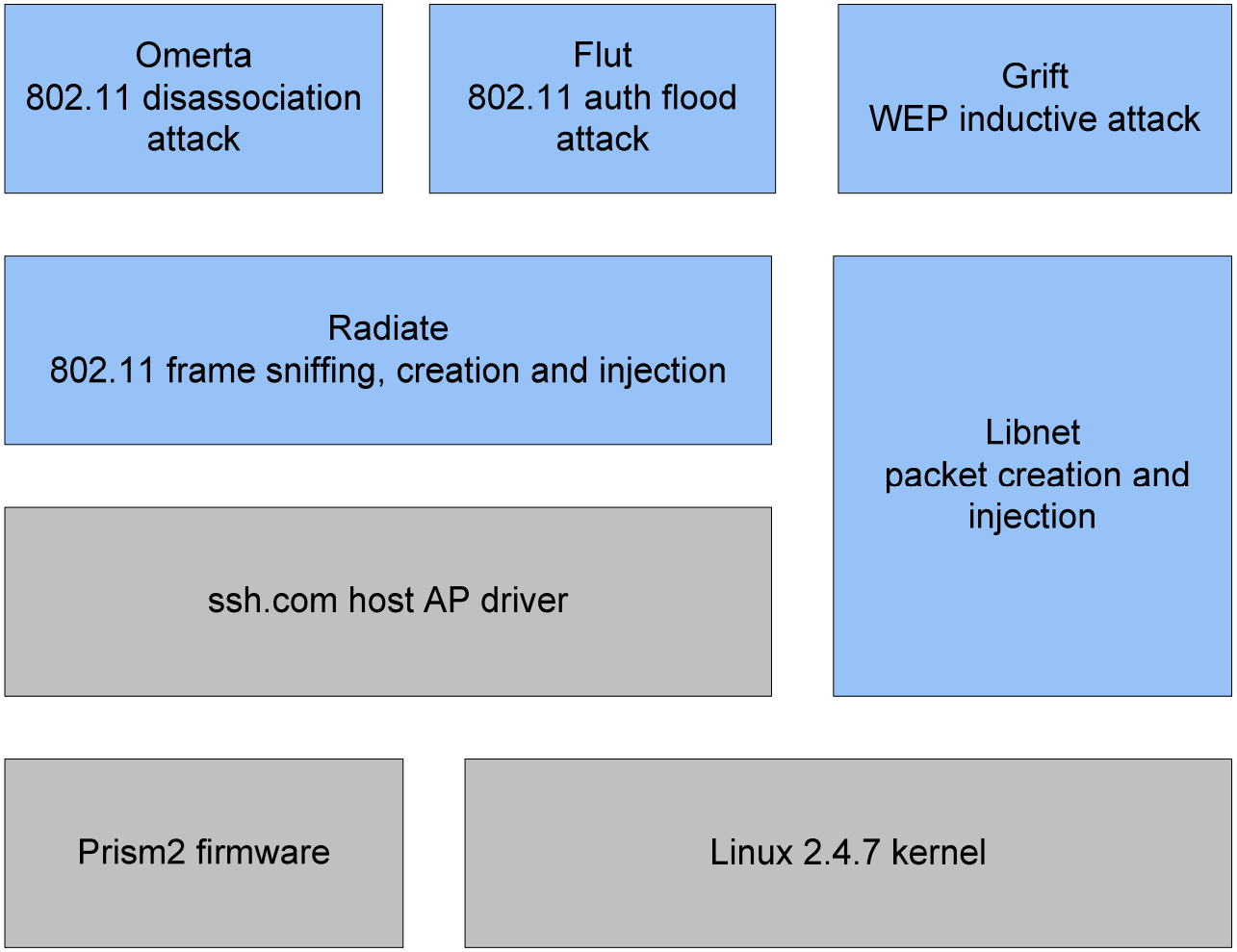
Radiate

- **The wireless analog to Libnet**
- **802.11b frame capturing, creation and injection library**
- **Simple C library consisting of a few function calls and a header file (at this point)**
- **Currently runs only under Intersil Prism2-based cards**
  - SMC, D-link, etc
  - Prism-2.5 support?

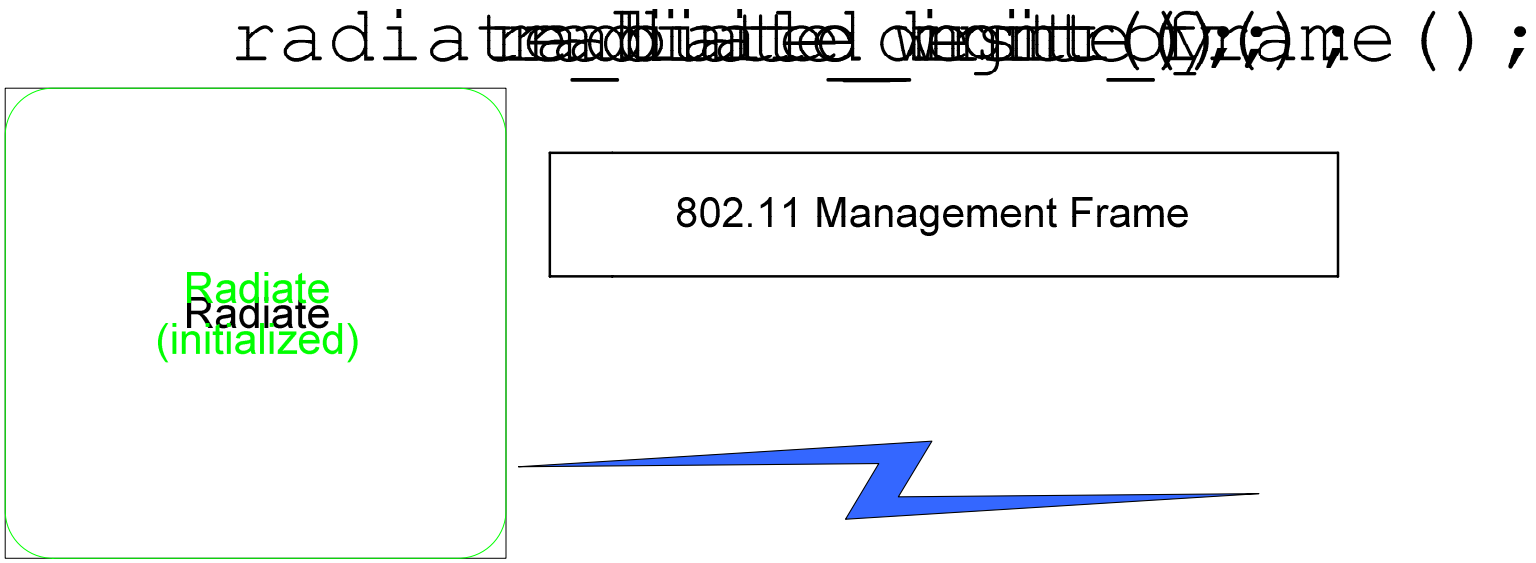
Radiate 50,000 foot view



Radiate and related component relationships



Radiate 25,000 foot view



Radiate initialization

- Radiate context
- Maintains state

```
radiate_t *r;
u_char control_flags = 0;

r = radiate_init(control_flags, errbuf);
if (r == NULL)
{
    fprintf(stderr, “radiate_init(): %s”, errbuf);
}
```

Radiate frame capture

- Blocking read (`recv(2)`)
- Buffer contains an `hfa384x_rx_frame` header'd frame

```
u_char *buf;
int c;

c = radiate_read(&buf, r);
if (c == -1)
{
    fprintf(stderr, "radiate_read(): %s\n",
            radiate_geterror(r));
    goto bad;
}
```

Radiate data frame creation

- Implicit `malloc(3)`

```
u_char *buf;
u_char data = “packet payload”;

buf = radiate_build_data_frame(mac1, mac2, bssid,
        RADIATE_DATA_STYPE_DATA, 0, data, strlen(data),
        r);
if (buf == NULL)
{
    fprintf(stderr, “radiate_build_data_frame(): %s\n”,
            radiate_geterror(r));
    goto bad;
}
```

Radiate management frame creation

- Implicit `malloc(3)`

```
u_char *buf;
u_char data = “packet payload”;

buf = radiate_build_mgmt_frame(mac1, mac2, bssid,
        RADIATE_MGMT_STYPE_AUTH, 0, data, strlen(data),
        r);
if (buf == NULL)
{
    fprintf(stderr, “radiate_build_mgmt_frame(): %s\n”,
            radiate_geterror(r));
    goto bad;
}
```

Radiate frame injection

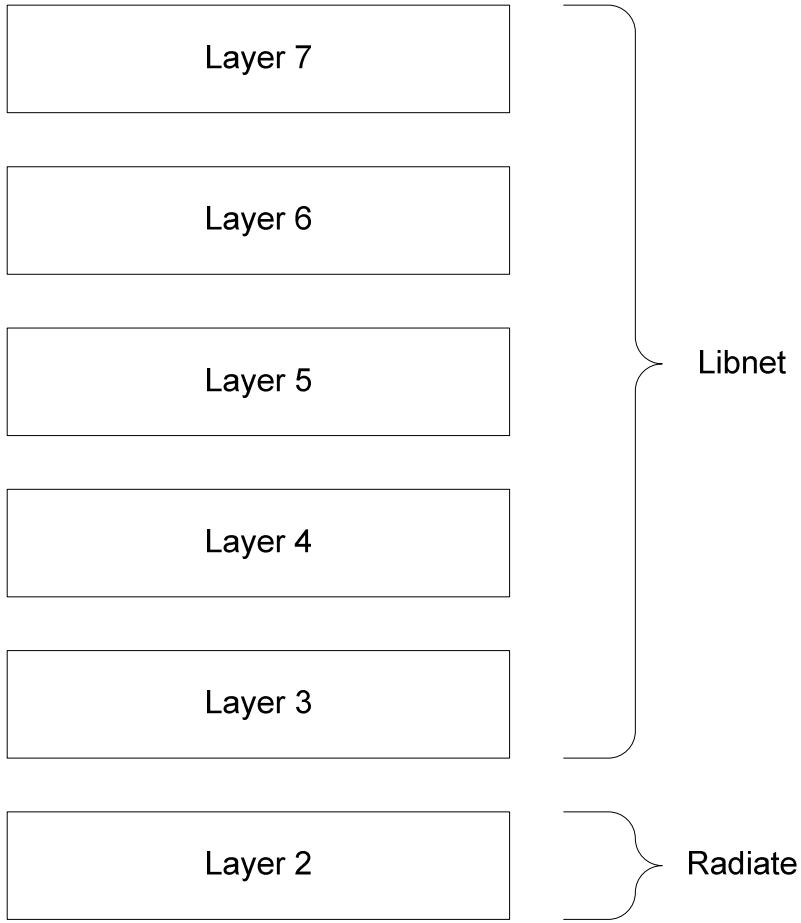
- via `send(2)`

```
int c;


c = radiate_write(frame, frame_s, r);
if (c == -1)
{
    fprintf(stderr, “radiate_write(): %s\n”,
            radiate_geterror(r));
    goto bad;
}
```

Radiate and Libnet – two best pals!

- Use Libnet to build layer 3 and above packet headers
- The Arbaugh Inductive attack was implemented with Radiate and Libnet



Radiate and Libnet Code Sample

```
u_char *packet;         /* wire-ready ICMP packet will be stored here */

u_long packet_s;        /* ICMP packet size will be stored here */


libnet_build_icmpv4_echo(ICMP_ECHO, 0, 0, 242, seq, NULL, 0, l, 0);

libnet_build_ipv4(28, 0, 242, 0, 64, IPPROTO_ICMP, 0, src, dst, NULL, 0, l,
   0);

/* XXX – You should never expose this pblock interface, idiot. */

libnet_pblock_coalesce(l, &packet, &packet_s);

frame = radiate_build_data_frame(a1, a2, bssid, RADIATE_DATA_STYPE_DATA, 0,
   packet, packet_s, r);

radiate_write(frame, frame_s, r);
```

Typical Radiate Program Usage

- **Make sure the library is built and installed. Duh.**
- **Pop your Prism-2 card in… Wait for the DooDeet.**
  - If you don't hear a DooDeet there is a driver / PCMCIA issue. But you're very smart so figuring it out will be a snap!
- **Put the card into monitor mode:**
  - `mkultra:~/Libradiate-0.2/scripts# ./set_monitor 1`
- **Set the channel you want to do your work on:**
  - `mkultra:~/Libradiate-0.2/scripts# ./set_channel 11`
- **Do it up!**
  - `mkultra:~/Code/802.11/Tools/Omerta# ./omerta`

By the way...

- **You don't get Omerta.**
- **You do get Radiate:**
  - http://www.packetfactory.net/Radiate

Oh, just something interesting to think about…

Known Issues

- We're at the mercy of the firmware…
- Problems with sending certain frames
  - To send data frames with the TO_DS bit set requires the module to be recompiled with `-DPRISM2_MONITOR_PACKET_INJECT`
    - Can't sniff frames when driver is built in this mode
  - Would be nice if we could make the card act as a passive radio
- Driver tends to crash or fail when card is removed and reseated frequently
- We need a logo ☹

Futures

- **This is pretty rough – we need a lot of code cleanup**
  - Buffer management (pblocks?)
  - API change to accommodate different layer 1 interfaces
- **We'll support additional cards**
  - Current issues with Prism-2 cards might go away
- **Did someone say Libnet Merger? Could be!**

Summary

- **The need exists to be able to develop arbitrary tools to test 802.11 networks for anomalous events such as Byzantine failure susceptibility**
- **Radiate is a toolkit that allows the application programmer to develop these tools**
- **802.11b frame handling**
  - Prism2 chipset
  - Aironet…?
- **Extra Special thanks to Timothy “The Newsh” Newsham**

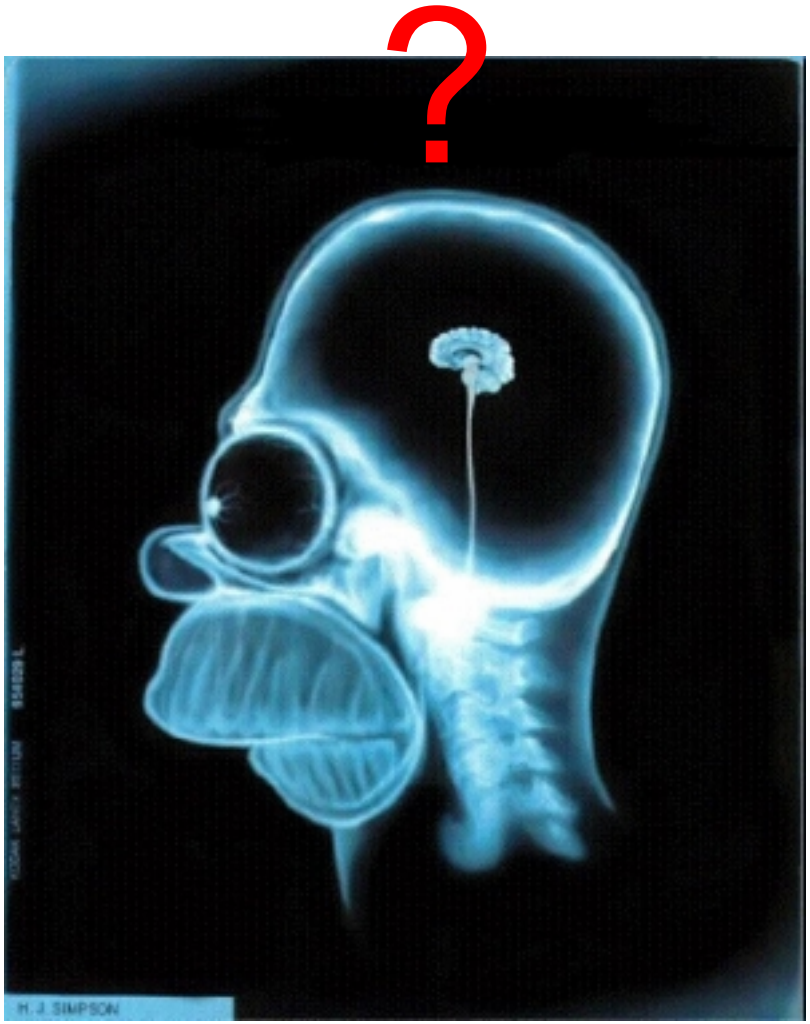
Building Open Source Network Security Tools

- **Simple C library; A "component"**
  - Upon which techniques are built
    - From which tools are created
- **New book on how to rapidly develop your own network security tools**
- **New paradigm for describing a network security tool; accelerates conceptualization and development**
- **Wiley & Sons**
- **Due out in October 2002**
- **Buy it!**

Questions?

Mike Schiffman

mike@atstake.com



Dave Pollino

dpollino@atstake.com