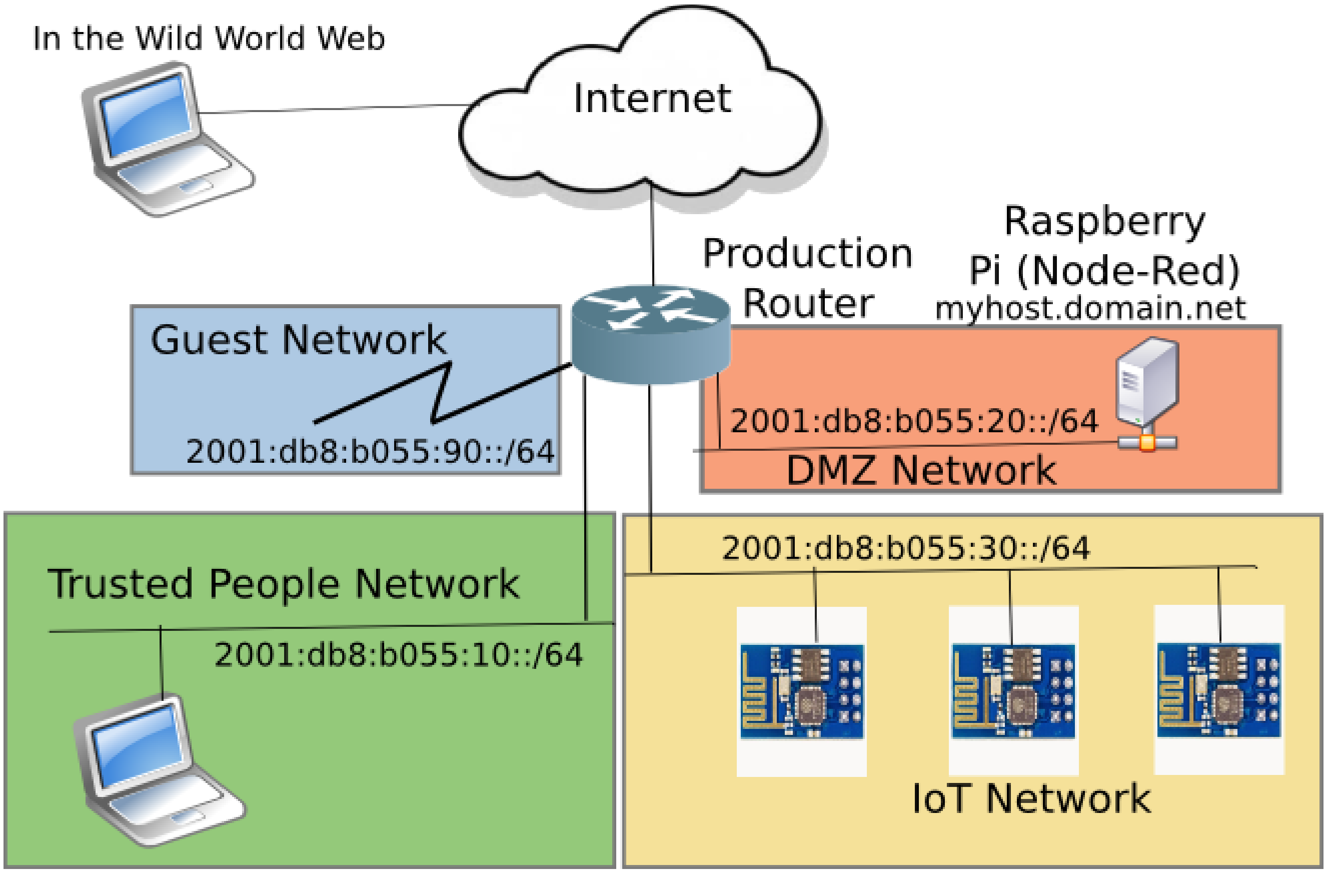
Flashy Lights and Security

Design the Network with Security in Mind



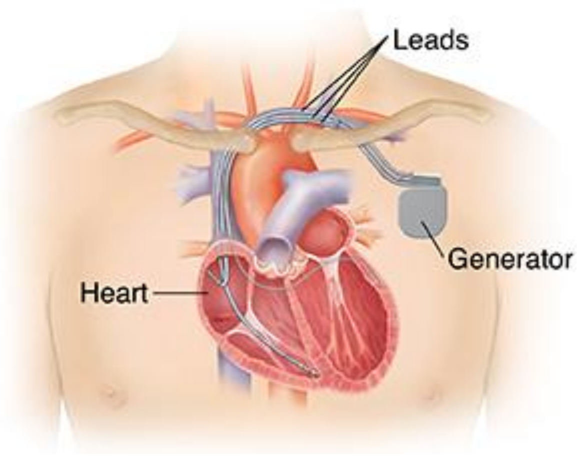
-
- Separate Networks, keeping the bad guys out, and the IoT network honest
- Set up Firewall rules (on the router) to enforce the following:
    - The **DMZ network** can be accessed from the internet
    - The **IoT Network** can **ONLY** connect to the server in the **DMZ network**!
    - The **Trusted People Network** can go anywhere
    - The **Guest Network**, you have friends, right?
- **Don't forget IPv6 and firewall rules**

HTTP vs HTTPS

- HTTP all data is transferred in plain text
- HTTPS all data is encrypted using public private key (PPK) encryption

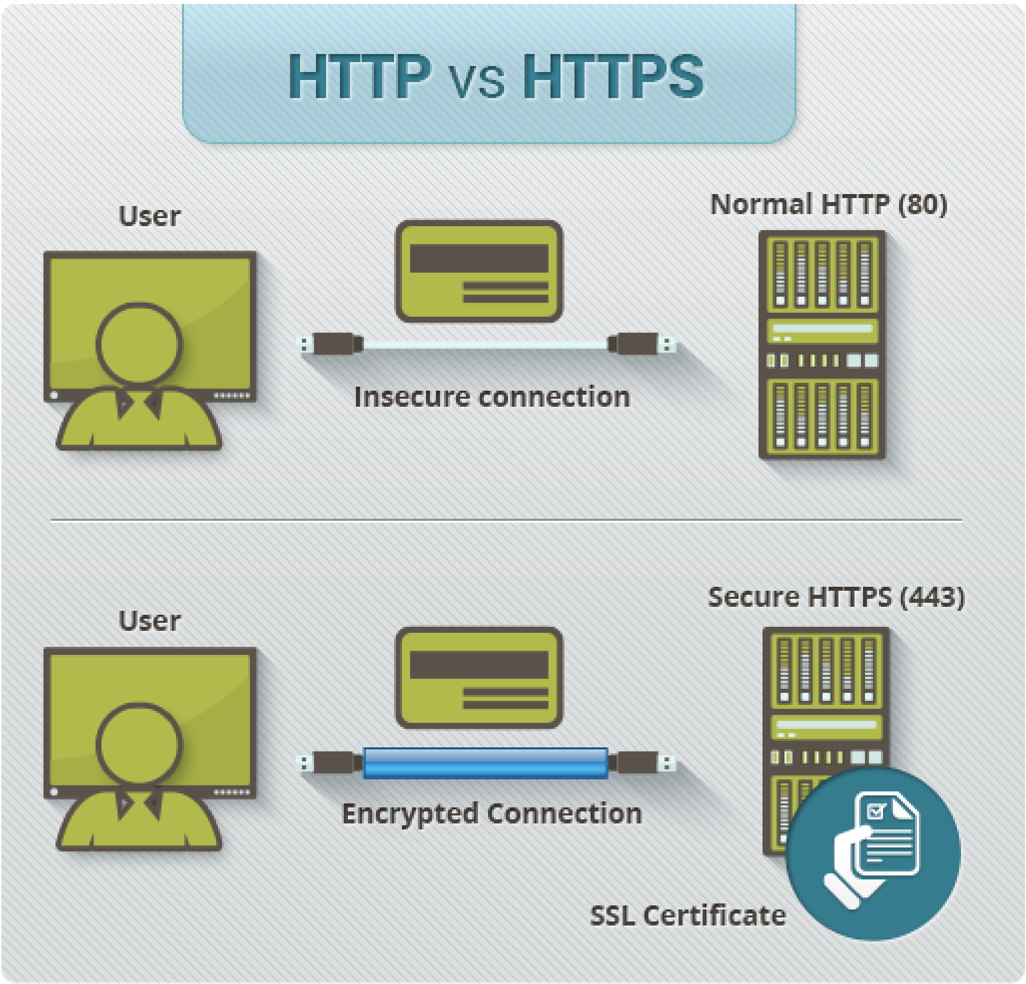
- For our application - changing flashing lights there is no need to encrypt the data.
    - http://leds/program?s=6 is **astonishingly** uninteresting but what is interesting is
    - the user id and password. These we want to keep secret. Both in this case and - wouldn't want the NSA messing with our lights :(
    - More importantly we wouldn't want the black hat hackers messing with our

- pacemakers, defibrillators, ponds and predator drones



So we want to ensure that only authorized folks can get in. Ways to do this:

1. ID, Password
2. PPK and passphrase where only authorized folks hold the private key and passphrase.
3. One time pad
4. ...

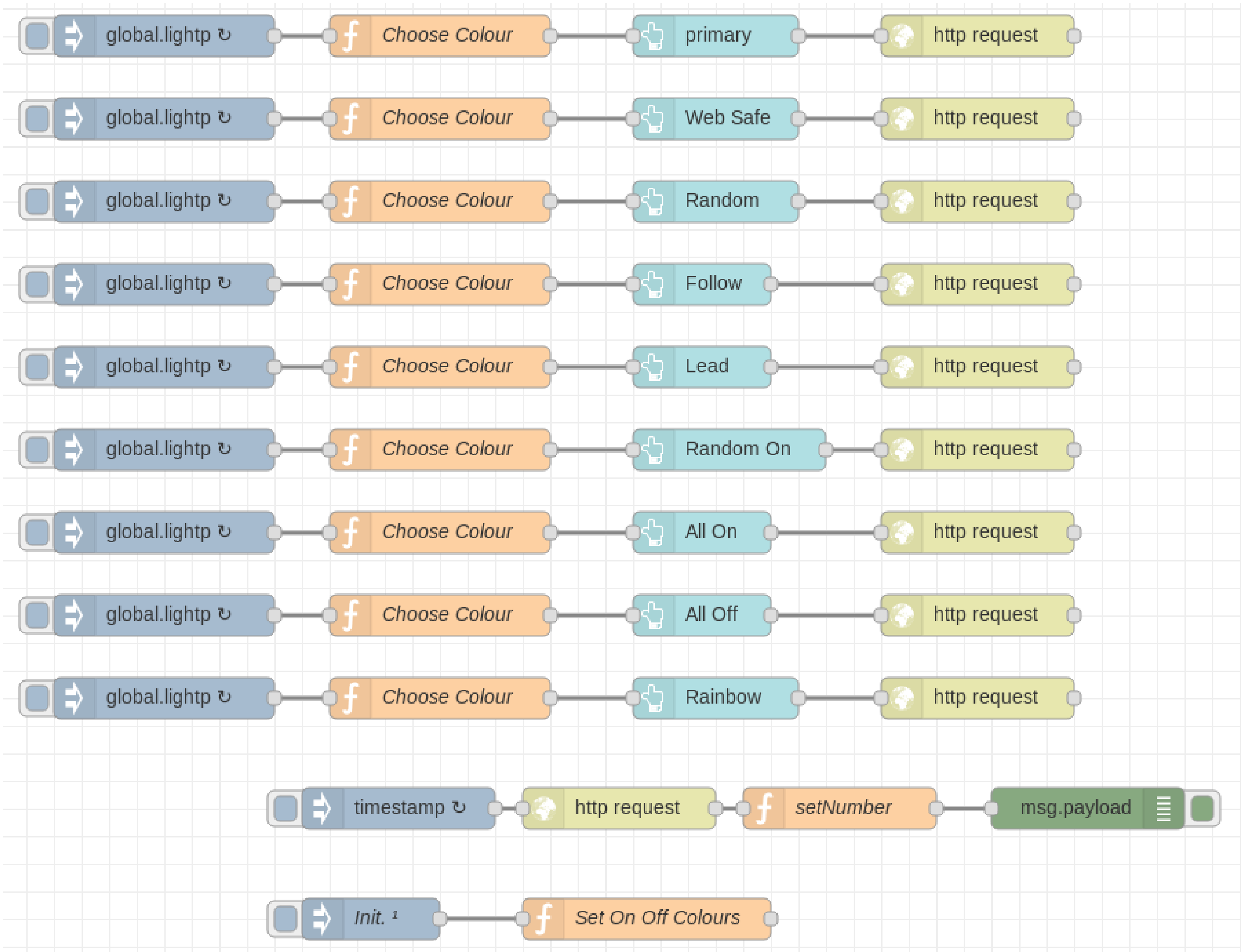


Node-Red on the Pi

- **Installation of node red**
    - Look at http://drsol.com/~deid/pi/leds2/Node-RED/index.html#InstallNode-RED for instructions on installing Node-RED on the Pi.

Connecting to the Node-RED server on your Pi:

- https://<ip address or hostname>:1880
- Description of flows



- **https://10.10.45.???**
    - **or**
- **https://[fd11::45]**

Choose Colour

```
var lightProgram = global.get('lightp') || 0;

if (lightProgram == 0 ){
    msg.background = global.get('onColour');
}else{
    msg.background = global.get('offColour');
}
return msg;
```

http request

```
http://pl9823:neopixelnot@leds/program?s=2
```

- **Creating a Secure Certificate with Let'sEncrypt and certbot**
  - Follow tutorial to install '**certbot**" for LetsEncrypt on the Pi
    - Do not need to compile binaris, as Pi 3 is an ARM7 system
      Added keys for debian jessie-backports
      ```
      > sudo apt-key adv --keyserver
      keyserver.ubuntu.com --recv-keys
      8B48AD6246925553
      > sudo apt-key adv --keyserver
      keyserver.ubuntu.com --recv-keys
      7638D0442B90D010
      ```
    - Then do
      ```
      apt-get install certbot
      ```
  - Run Certbot on the Pi
    - **Note**: Pi must be running a webserver, and it **must** be accessible to the outside world (adjust our firewall as needed)

    -
      ```
      sudo certbot certonly --webroot -w
      /var/www/html --email admin@domain.net
      --rsa-key-size 4096 -d
      myhost.domain.net

      Saving debug log to
      /var/log/letsencrypt/letsencrypt.log
      Obtaining a new certificate
      Performing the following challenges:
      http-01 challenge for
      6rasport.cvmiller.net
      Using the webroot path /var/www/html
      for all unmatched domains.
      Waiting for verification...
      Cleaning up challenges
      Generating key (4096 bits):
      /etc/letsencrypt/keys/0000_key-
      certbot.pem
      Creating CSR:
      /etc/letsencrypt/csr/0000_csr-
      ```

```
        certbot.pem
        Cert is stored in /etc/letsencrypt/live/myhost.domain.net/
        ```

    - then copy the private and fullchain certs (pem files) to /home/pi/.node-red directory

        -
          ```
          sudo cp /etc/letsencrypt/live/6rasport.cvmiller.net/privkey.pem /home/pi/.node-red/privatekey.pem
          sudo cp /etc/letsencrypt/live/6rasport.cvmiller.net/fullchain.pem /home/pi/.node-red/certificate.pem
          ```
    - Edit node-red settings.js file
        - Uncomment and add the names of the certificate file you created a while ago.

            -
              ```
              // Add Certs
                  https: {
                      key: fs.readFileSync('/home/pi/.node-red/privatekey.pem'),
                      cert: fs.readFileSync('/home/pi/.node-red/certificate.pem')
                  },
              ```
    - Restart node-red
    -
- **Creating and installing a home brewed secure certificate**
    - cd to your home directory then cd to .node-red and execute the following commands to create a self signed certificate
            - openssl genrsa -out privatekey.pem 1024
            - openssl req -new -key privatekey.pem -out private-csr.pem
            - openssl x509 -req -days 3650 -in private-csr.pem -signkey privatekey.pem -out certificate.pem
    - Modify the node-red config file settings.js
    - Uncomment the line: `var fs = require("fs");`
    - Uncomment the lines and add credentials (see below):

        -
          ```
          // Deid uncommented
              adminAuth: {
                  type: "credentials",
                  users: [{
          ```

```
                username: "deid",
                password: "$2a$08$etA.ES98hxaVl9zYQN8xX.uAlQT/rRuARzdokGr7SmuGCog77jYPm",
                permissions: "*"
            }]
        },
```

    - Uncomment and add credentials:

        -
          ```
          // Deid modified uncommented and insert creds. 20190909
              httpNodeAuth: {user:"deid",pass:"$2a$08$etA.ES98hxaVl9zYQN8xX.uAlQT/rRuARzdokGr7SmuGCog77jYPm"},
              httpStaticAuth: {user:"deid",pass:"$2a$08$etA.ES98hxaVl9zYQN8xX.uAlQT/rRuARzdokGr7SmuGCog77jYPm"},
          ```
    - Uncomment and add the names of the certificate file you created a while ago.

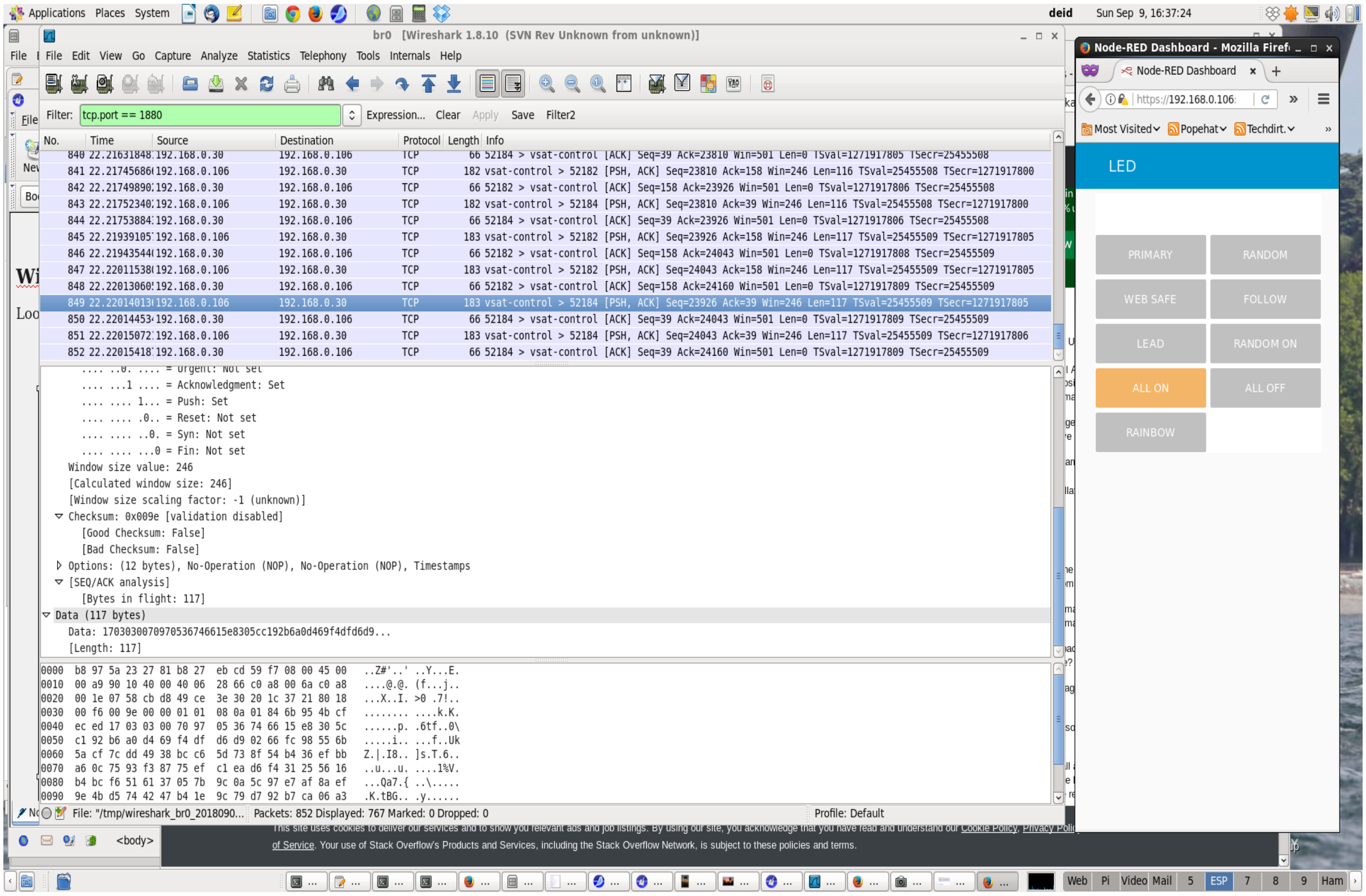
        -
          ```
          // Deid uncommented.
              https: {
                  key: fs.readFileSync('/home/pi/.node-red/privatekey.pem'),
                  cert: fs.readFileSync('/home/pi/.node-red/certificate.pem')
              },
          ```

- **Creating Credentials**
    - The credentials are a bcrypt hash. To generate a suitable password hash, you can use the `node-red-admin` command-line tool:
        - `node-red-admin hash-pw`
        - It will query for the password and return the bcrypt hash.

Wireshark

Looking at the data transmission

- https


Applications Places System
deid Sun Sep 9, 16:37:24
br0 [Wireshark 1.8.10 (SVN Rev Unknown from unknown)]
File Edit View Go Capture Analyze Statistics Telephony Tools Internals Help
Expression... Clear Apply Save Filter2
Node-RED Dashboard - Mozilla Firefox
Node-RED Dashboard
https://192.168.0.106
Most Visited Popehat Techdirt.
LED
PRIMARY
RANDOM
WEB SAFE
FOLLOW
LEAD
RANDOM ON
ALL ON
ALL OFF
RAINBOW
Profile: Default
<body>
Web Pi Video Mail 5 ESP 7 8 9 Ham

```
Filter: tcp.port == 1880

No.  Time         Source        Destination   Protocol Length Info
840 22.21631848 192.168.0.30   192.168.0.106  TCP      66 52184 > vsat-control [ACK] Seq=39 Ack=23810 Win=501 Len=0 TSval=1271917805 TSecr=25455508
841 22.21745686 192.168.0.106  192.168.0.30   TCP     182 vsat-control > 52182 [PSH, ACK] Seq=23810 Ack=158 Win=246 Len=116 TSval=25455508 TSecr=1271917800
842 22.21749890 192.168.0.30   192.168.0.106  TCP      66 52182 > vsat-control [ACK] Seq=158 Ack=23926 Win=501 Len=0 TSval=1271917806 TSecr=25455508
843 22.21752340 192.168.0.106  192.168.0.30   TCP     182 vsat-control > 52184 [PSH, ACK] Seq=23810 Ack=39 Win=246 Len=116 TSval=25455508 TSecr=1271917800
844 22.21753884 192.168.0.30   192.168.0.106  TCP      66 52184 > vsat-control [ACK] Seq=39 Ack=23926 Win=501 Len=0 TSval=1271917806 TSecr=25455508
845 22.21939105 192.168.0.106  192.168.0.30   TCP     183 vsat-control > 52182 [PSH, ACK] Seq=23926 Ack=158 Win=246 Len=117 TSval=25455509 TSecr=1271917805
846 22.21943544 192.168.0.30   192.168.0.106  TCP      66 52182 > vsat-control [ACK] Seq=158 Ack=24043 Win=501 Len=0 TSval=1271917808 TSecr=25455509
847 22.22011538 192.168.0.106  192.168.0.30   TCP     183 vsat-control > 52182 [PSH, ACK] Seq=24043 Ack=158 Win=246 Len=117 TSval=25455509 TSecr=1271917805
848 22.22013060 192.168.0.30   192.168.0.106  TCP      66 52182 > vsat-control [ACK] Seq=158 Ack=24160 Win=501 Len=0 TSval=1271917809 TSecr=25455509
849 22.22014013 192.168.0.106  192.168.0.30   TCP     183 vsat-control > 52184 [PSH, ACK] Seq=23926 Ack=39 Win=246 Len=117 TSval=25455509 TSecr=1271917805
850 22.22014453 192.168.0.30   192.168.0.106  TCP      66 52184 > vsat-control [ACK] Seq=39 Ack=24043 Win=501 Len=0 TSval=1271917809 TSecr=25455509
851 22.22015072 192.168.0.106  192.168.0.30   TCP     183 vsat-control > 52184 [PSH, ACK] Seq=24043 Ack=39 Win=246 Len=117 TSval=25455509 TSecr=1271917806
852 22.22015418 192.168.0.30   192.168.0.106  TCP      66 52184 > vsat-control [ACK] Seq=39 Ack=24160 Win=501 Len=0 TSval=1271917809 TSecr=25455509
```

```
      .... ..0. .... = Urgent: Not set
      .... ...1 .... = Acknowledgment: Set
      .... .... 1... = Push: Set
      .... .... .0.. = Reset: Not set
      .... .... ..0. = Syn: Not set
      .... .... ...0 = Fin: Not set
    Window size value: 246
    [Calculated window size: 246]
    [Window size scaling factor: -1 (unknown)]
  ▽ Checksum: 0x009e [validation disabled]
      [Good Checksum: False]
      [Bad Checksum: False]
  ▷ Options: (12 bytes), No-Operation (NOP), No-Operation (NOP), Timestamps
  ▽ [SEQ/ACK analysis]
      [Bytes in flight: 117]
▽ Data (117 bytes)
    Data: 1703030070970536746615e8305cc192b6a0d469f4dfd6d9...
    [Length: 117]
```

```
0000  b8 97 5a 23 27 81 b8 27  eb cd 59 f7 08 00 45 00   ..Z#'..' ..Y...E.
0010  00 a9 90 10 40 00 40 06  28 66 c0 a8 00 6a c0 a8   ....@.@. (f...j..
0020  00 1e 07 58 cb d8 49 ce  3e 30 20 1c 37 21 80 18   ...X..I. >0 .7!..
0030  00 f6 00 9e 00 00 01 01  08 0a 01 84 6b 95 4b cf   ........ ....k.K.
0040  ec ed 17 03 03 00 70 97  05 36 74 66 15 e8 30 5c   ......p. .6tf..0\
0050  c1 92 b6 a0 d4 69 f4 df  d6 d9 02 66 fc 98 55 6b   .....i.. ...f..Uk
0060  5a cf 7c dd 49 38 bc c6  5d 73 8f 54 b4 36 ef bb   Z.|.I8.. ]s.T.6..
0070  a6 0c 75 93 f3 87 75 ef  c1 ea d6 f4 31 25 56 16   ..u...u. ....1%V.
0080  b4 bc f6 51 61 37 05 7b  9c 0a 5c 97 e7 af 8a ef   ...Qa7.{ ..\.....
0090  9e 4b d5 74 42 47 b4 1e  9c 79 d7 92 b7 ca 06 a3   .K.tBG.. .y......
```

File: "/tmp/wireshark_br0_2018090... Packets: 852 Displayed: 767 Marked: 0 Dropped: 0

This site uses cookies to deliver our services and to show you relevant ads and job listings. By using our site, you acknowledge that you have read and understand our Cookie Policy, Privacy Policy, and our Terms of Service. Your use of Stack Overflow's Products and Services, including the Stack Overflow Network, is subject to these policies and terms.

- From /etc/services:
  - vsat-control   1880/tcp   # Gilat VSAT Control
  - vsat-control   1880/udp   # Gilat VSAT Control

- We are NOT doing using port 1880 for vsat control. It turns out that 1880 is also the default port for Node-red.

Whats interesting here, is that nothing is interesting.

IPV6

- Although IPv6 may seem scary, it isn't. One may assume that it is *Just Like IPv4*, but it isn't. IPv6 is a different protocol that still uses TCP and UDP for transport. IPv6 has its own firewall rules, and routing protocols. That said, one can learn the basics of IPv6 very quickly if they understand how networks work (with IPv4).

Why use IPv6?

- **Because it is easier**. There is no NAT (Net Address Translations) in the way, and correspondingly, no port forwards to set up. Your application is simpler. IPv6 eliminates the evils of NAT
  - **Broken connectivity** - The many to one nature of NAT breaks any to any connectivity originally conceived by the founders of the internet
  - **Simplifies Communications** - Other traversal protocols like STUN (Session Traversal Utilities for NAT), TURN (Traversal Using Relays around NAT), IGDP (UPnP Internet Gateway Device Protocol), and ALGs (Application-level gateway) are no longer needed
  - **Easier Troubleshooting** - Since the client address is the real address, and one doesn't have to look through one or more NAT devices to see the mapping of addresses just to troubleshoot a path

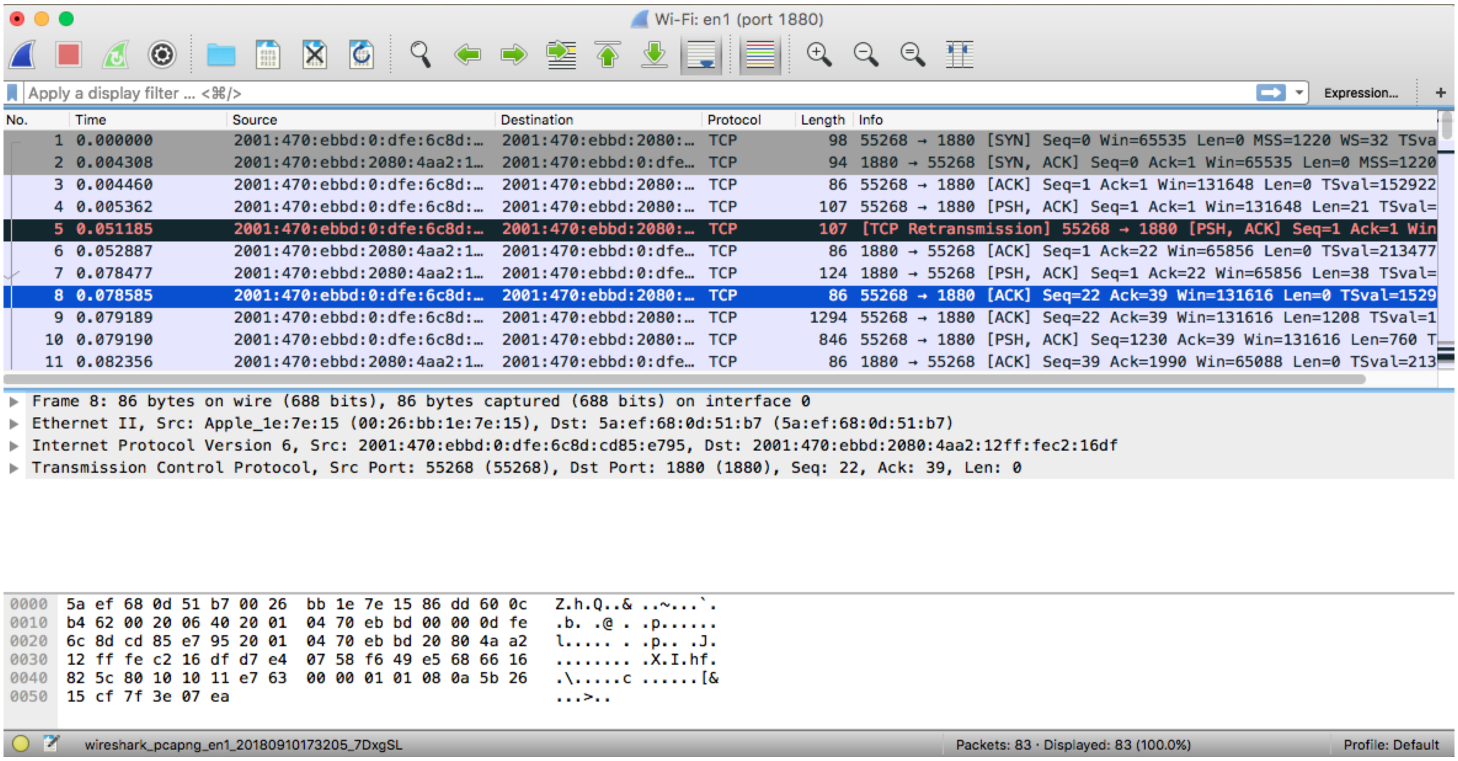
- **Because you will be future-ready**. The world has run out of IPv4 addresses (a couple of years ago), but the internet continues to grow. It can only grow using a larger address space.

- **It is already deployed on every PC**, Mac, and smart phone. In fact, there are wireless providers such as T-Mobile who are running IPv6-only networks with over 20 million users on their network. According to Google Stats, **over 20%** of internet traffic today is over IPv6.

Using host names rather than IP addresses

- Utilizing IPv6 should be transparent to the user. Rather than using a literal IPv4 address, such as 10.10.45.10, or an IPv6 address such as FD11::45, a better practice is to use a host name. Host names utilize the abstraction layer of DNS. If you don't have control of DNS, then put the host name into /etc/hosts
  - `mypi fd11::45`
- Once a host name has been setup, then the name can be used in the brower URL like any other website:
  - `https://mypi:1880/`
- Using a host name, also gets around the difficult issue of colons. In the IPv4 world, colons usually represent a specific TCP (or UDP) port, e.g.
  - `10.10.45.10:1880`
- But IPv6 uses colons to demark the Quibbles of the IPv6 address. So a rather cludgy way was invented to indicate the port number with an ipv6 address:
  - `[fd11::45]:1880`
- As you can see it is much easier to use host names:
  - `mypi:1880`

- When sniffing an IPv6 transaction using wireshark, it is often easy to use the following capture filter which will filter out all that IPv4 cruft:
  - ip6



```
No.  Time      Source                     Destination             Protocol Length Info
   1 0.000000  2001:470:ebbd:0:dfe:6c8d:…  2001:470:ebbd:2080:…  TCP        98 55268 → 1880 [SYN] Seq=0 Win=65535 Len=0 MSS=1220 WS=32 TSva
   2 0.004308  2001:470:ebbd:2080:4aa2:1…  2001:470:ebbd:0:dfe…  TCP        94 1880 → 55268 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1220
   3 0.004460  2001:470:ebbd:0:dfe:6c8d:…  2001:470:ebbd:2080:…  TCP        86 55268 → 1880 [ACK] Seq=1 Ack=1 Win=131648 Len=0 TSval=152922
   4 0.005362  2001:470:ebbd:0:dfe:6c8d:…  2001:470:ebbd:2080:…  TCP       107 55268 → 1880 [PSH, ACK] Seq=1 Ack=1 Win=131648 Len=21 TSval=
   5 0.051185  2001:470:ebbd:0:dfe:6c8d:…  2001:470:ebbd:2080:…  TCP       107 [TCP Retransmission] 55268 → 1880 [PSH, ACK] Seq=1 Ack=1 Win
   6 0.052887  2001:470:ebbd:2080:4aa2:1…  2001:470:ebbd:0:dfe…  TCP        86 1880 → 55268 [ACK] Seq=1 Ack=22 Win=65856 Len=0 TSval=213477
   7 0.078477  2001:470:ebbd:2080:4aa2:1…  2001:470:ebbd:0:dfe…  TCP       124 1880 → 55268 [PSH, ACK] Seq=1 Ack=22 Win=65856 Len=38 TSval=
   8 0.078585  2001:470:ebbd:0:dfe:6c8d:…  2001:470:ebbd:2080:…  TCP        86 55268 → 1880 [ACK] Seq=22 Ack=39 Win=131616 Len=0 TSval=1529
   9 0.079189  2001:470:ebbd:0:dfe:6c8d:…  2001:470:ebbd:2080:…  TCP      1294 55268 → 1880 [ACK] Seq=22 Ack=39 Win=131616 Len=1208 TSval=1
  10 0.079190  2001:470:ebbd:0:dfe:6c8d:…  2001:470:ebbd:2080:…  TCP       846 55268 → 1880 [PSH, ACK] Seq=1230 Ack=39 Win=131616 Len=760 T
  11 0.082356  2001:470:ebbd:2080:4aa2:1…  2001:470:ebbd:0:dfe…  TCP        86 1880 → 55268 [ACK] Seq=39 Ack=1990 Win=65088 Len=0 TSval=213
```

```
▶ Frame 8: 86 bytes on wire (688 bits), 86 bytes captured (688 bits) on interface 0
▶ Ethernet II, Src: Apple_1e:7e:15 (00:26:bb:1e:7e:15), Dst: 5a:ef:68:0d:51:b7 (5a:ef:68:0d:51:b7)
▶ Internet Protocol Version 6, Src: 2001:470:ebbd:0:dfe:6c8d:cd85:e795, Dst: 2001:470:ebbd:2080:4aa2:12ff:fec2:16df
▶ Transmission Control Protocol, Src Port: 55268 (55268), Dst Port: 1880 (1880), Seq: 22, Ack: 39, Len: 0
```

```
0000  5a ef 68 0d 51 b7 00 26  bb 1e 7e 15 86 dd 60 0c   Z.h.Q..& ..~...`.
0010  b4 62 00 20 06 40 20 01  04 70 eb bd 00 00 0d fe   .b. .@ . .p......
0020  6c 8d cd 85 e7 95 20 01  04 70 eb bd 20 80 4a a2   l..... . .p.. .J.
0030  12 ff fe c2 16 df d7 e4  07 58 f6 49 e5 68 66 16   ........ .X.I.hf.
0040  82 5c 80 10 10 11 e7 63  00 00 01 01 08 0a 5b 26   .\.....c ......[&
0050  15 cf 7f 3e 07 ea                                  ...>..
```

wireshark_pcapng_en1_20180910173205_7DxgSL   Packets: 83 · Displayed: 83 (100.0%)   Profile: Default

- As you can see, the capture looks very similar to the IPv4 wireshark capture earlier, but with IPv6 addresses.
- Whats interesting here, is that nothing is interesting. IPv6 should be transparent to the user.

Node-RED and IPV6

- Modification to ../.node-red/settings.js to listen on IPV6.

```
    // By default, the Node-RED UI accepts connections on all IPv4 interfaces.
    // Deid added IPv6 stuff.
    // To listen on all IPv6 addresses, set uiHost to "::",
    // The following property can be used to listen on a specific interface. For
    // example, the following would only allow connections from the local machine.
    //uiHost: "127.0.0.1",
    uiHost: "::",
```

Notes:

- Certificates are by domain-name only
  - By hijacking DNS, we use local addresses, rather than the real (internet) addresses
- Certbot **Requires** Internet connectivity
  - It creates **Challenges**, and places them in the Document-Root of the webserver in order to prove to LetsEncrypt that the Cert is being generated for the correct domain-name