Attacking The Internet of Things (using time)

Paul McMillan

Who am I?

There are many ways
to attack the
Internet of Things

Demo
(start)

What is a timing attack?

```
def authenticate_user(user, password):
    stored_hash=get_password_hash(user):
    if stored_hash:
        test_hash = sha1(password)
        if test_hash == stored_hash:
            Return True
    Return False
```

Many Kinds

- User Enumeration
- Blind SQL Injection
- CPU Cache attacks against crypto
  - Local
  - Cross-VM
- Lucky 13
- Many more...

String Comparison Timing Attacks

memcmp

```
while (len != 0)
    {
      a0 = ((byte *) srcp1)[0];
      b0 = ((byte *) srcp2)[0];
      srcp1 += 1;
      srcp2 += 1;
      res = a0 - b0;
      if (res != 0)
         return res;
      len -= 1;
    }
```

ABCD

AXXX

BXXX

CXXX

DXXX

ABCD

AAXX

ABXX

ACXX

ADXX

ABCD

ABAX

ABBX

ABCX

ABDX

ABCD

ABCA
ABCB
ABCC
ABCD

MASSIVE Speedup

c = character set

n = Length of string

Brute Force: c^n

Timing Attack: c * n (* x)

(x is # tries to distinguish)

Why are they interesting?

What are the drawbacks?

Let's talk about time

Internet SF-NY 70ms
Spinning Disk 13ms

Ram Latency 83ns
L1 cache 1ns
1 cpu cycle ~0.33ns

Speed of light
in network cable

1 meter in ~5ns
200 meters ~1µs

So… how long does each byte of that string comparison take?

nanoseconds

(on a modern 3Ghz machine)

What about something a little slower?


PHILIPS
I WANT / JE VEUX
hue
Available on the App Store
PHILIPS
I WANT / JE VEUX
hue

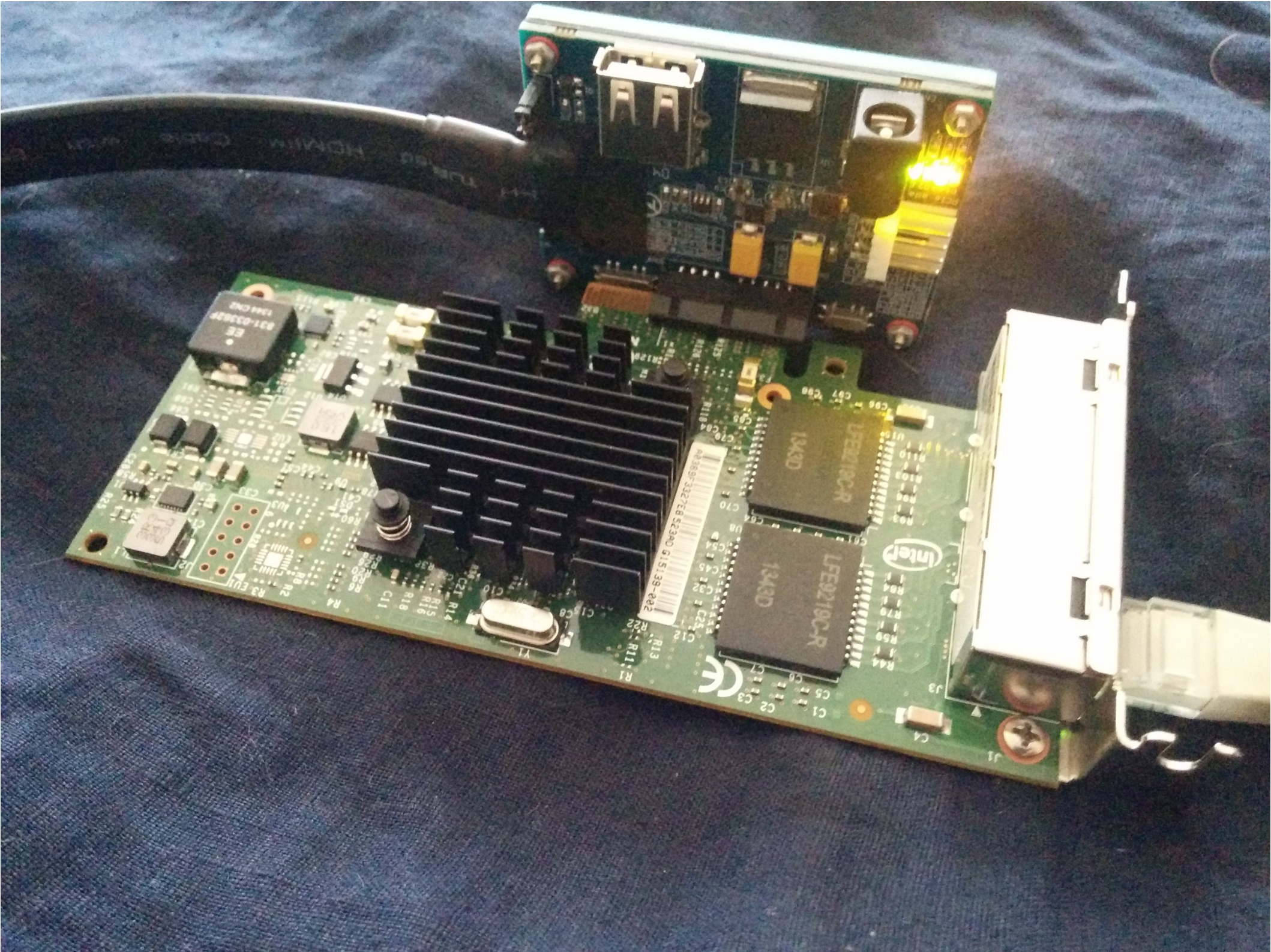
Network timing precision

Sources of Imprecision

- Graphics drivers
- Background networking
- USB Devices
- Power saving measures
- Audio devices
- Etc.

Software Timestamps are noisy.

Let's use hardware!


LFE9218CR
1343D
LFE9218CR
1343D
intel

Data Collection

- Generate repeated traffic
- TCPdump the packets
- Analyze the data
- Feed back to traffic gen

Making things work

- Libpcap 1.5.0+
- TCPDump 4.6.0+ (released July 2, 2014)
- Recent-ish Kernel

Compile these from source.

In theory, this might work on OSX?

It works on Ubuntu 14.04 for me.

Nanoseconds. Ugh!

- Scapy doesn't read the pcap files
- Neither do most other packages
- Wireshark does! (so does my fork of dpkt)

- Nanosecond timestamps lose precision if you convert them to a float()
- So we subtract a large offset, and don't work with raw timestamps.
- Use integer Nanoseconds rather than float seconds

What is the Hue API?

- GET /api/<user token>/lights
- Basic RESTful API
- Not very smart - always returns http status 200 even when returning errors.
- User token is the only required auth

  (no username, no sessions)
- Not very fast (can handle ~30-60req/s)

Hue Base Oddities

- Running FreeRTOS/6.0.5
- Network stack is dumber than a sack of rocks
- SSDP implementation ignores most parameters
- Each HTTP response is split into ~8 pieces
- HTTP stack ignores most incoming headers
- Ethernet Frame Check Sequence sometimes wrong
- Noisy: Broadcasts SSDP, ARPs for OpenDNS

Statistics!

Basic Review

- What is the Null Hypothesis?
- What does it mean to reject the null hypothesis?

- What are we trying to do here?
  - sorting samples into groups, and identifying outlier

More Stats

- Why can't we use the t-test?
- What is the K-S test, and why does it help us here?
- What other approaches might we use?

Data Prep

- Discarding data (Look at your data, pick the cleanest part without clipping if you can)
- How to do that wrong!
- Why?

Code

- In the repo now, public after the talk:

  https://github.com/PaulMcMillan/2014_ekoparty
- several separate but related scripts
- Don't forget to save your data for re-analysis
- Starting points for analysis, not full blown attack tools

Tips and Tricks

Keep the socket open (if you can)

Configure your network card parameters properly

- use hardware timestamps
- turn off queues
- use nanosecond timestamps (gah!)
- Turn off some offloads

Make everything Quiet

- reduce extraneous traffic to the device
- Slow loris to exclude other traffic
- don't run extra stuff on your attack machine
- Profile your victim – discard noisy periods

Do a warmup run before starting to collect data!

Find the simplest possible request

Avoid statistical tests that assume your data is normal

Gather data on all hypothesis concurrently

Randomize the request order

Don't overwhelm the device

Don't forget you can stop and brute force a token

Some code short circuits if strings aren't the same length.

Try both:
Fast and Noisy
Slow and Quiet

Which one works best will vary.

Don't get fooled by your own data!

When in doubt, take more samples.

If you try hard enough, eventually ‘statistically significant’ findings will emerge from any reasonably complicated data set.

When results are analyzed many ways without a plan, the results simply cannot be interpreted.

At best, you can treat the findings as an hypothesis to be tested in future studies with new data.

Questions?
http://github.com/PaulMcMillan/2014_ekoparty

Repo contains slides and many useful references.

Presentation references in speaker notes.