Pcap-NG Logging: Packets and Packet-Like Objects

Mike Kershaw
infosec.exchange/@kismetwireless || https://www.kismetwireless.net

Hello!

- OSS developer (Kismet and other tools)
- Engineer/Hacker @ Hak5 for embedded products (WiFi Pineapple, Packet Squirrel, LAN Turtle, etc)
- Fairly long history in WiFi nonsense & working for WiFi vendors

Logging is Everything

- Logs are often the *only* source of information about an event
- Only way to time travel back and figure out what happened during an event
- The more complete the logs, the more useful they are
- This makes our goal pretty clear:

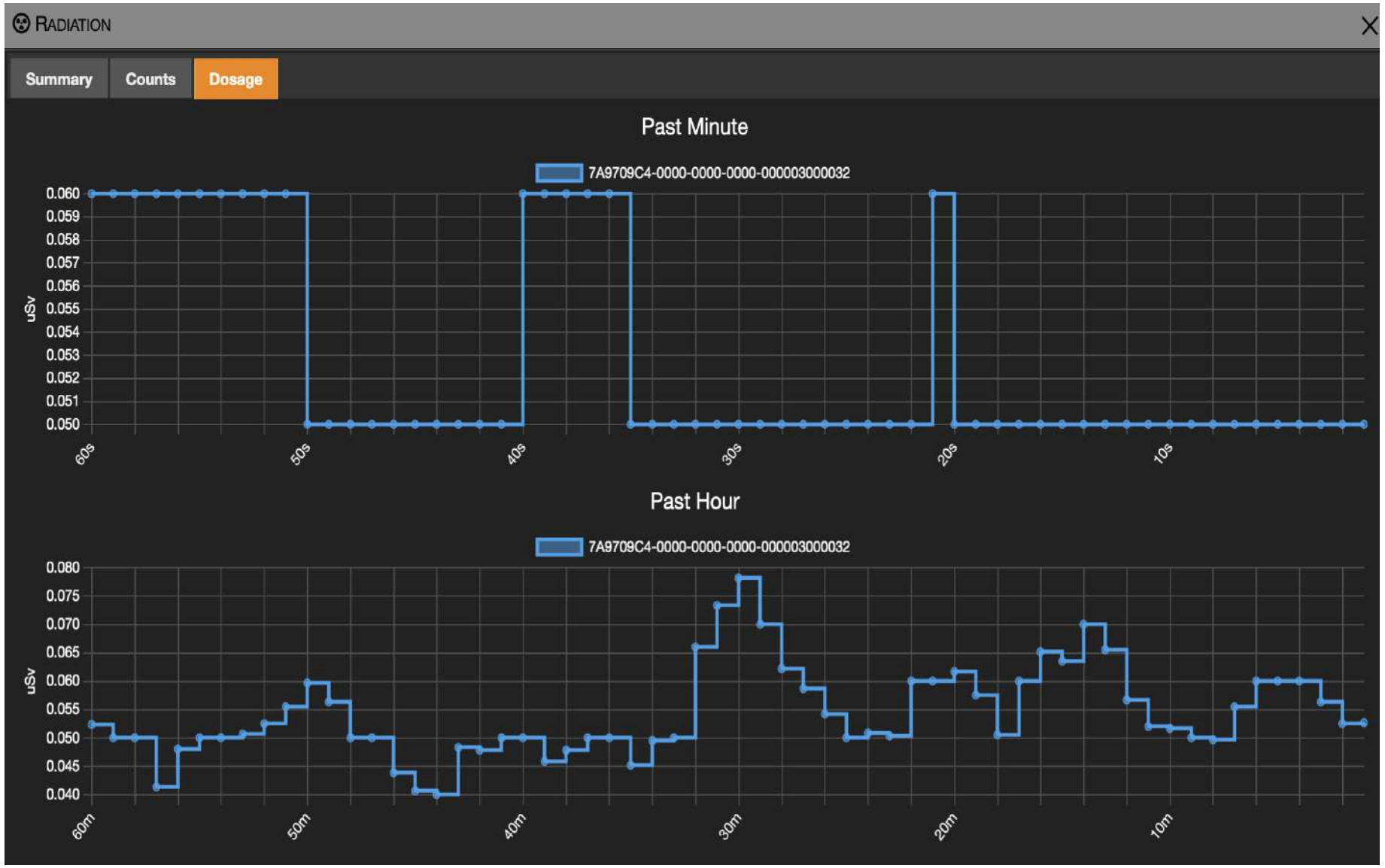
Capture everything possible

What is *Everything*?

- All (or as close to all) transmitting devices

- But also, metadata like…
- Location
- Speed
- Temperature and humidity
- Antenna attributes
- Orientation of the capture HW (elevation, azimuth)
- Anything else we can think of!

How Esoteric Do You Want To Be?



Wouldn't You, If You Could?

- If it were possible to capture as much about the external environment at the same time as packets, wouldn't you?
- Factors like temperature and humidity can be relevant while performing diagnostics and logging problems
- "The network goes out every time it's cold"

Traditional Pcap Format

- PCAP header
  - Time (milliseconds)
  - Link type (DLT)
  - Packet size
- Packets
  - Time (milliseconds)
  - Packet content
- All packets must be the same type
- All information must be in the packet

PPI

- Custom headers used in pcap
- PPI tried to solve for Wi-Fi
- Wrapper header includes signal levels, GPS location

- Limited signal information
- No way to indicate multiple interfaces
- Only for Wi-Fi

Radiotap

- Adopted in the kernel
- Wrapper header has complex Wi-Fi signal info
- No concept of location

- Still can’t mix multiple capture types, Wi-Fi only

Multiple Interfaces

- Why do we care about multiple interfaces & types?
- Radio in particular shares space with other protocols
- Performance of one can be impacted by traffic on others
- In a mixed environment, ignoring one can mask problems

Limits?

- How many interfaces can you put in a Pcap-NG?
- Technically unlimited…
- I know of users putting hundreds – to thousands – of capture interfaces in a single Pcap-NG

Pcap-NG

- To solve these and other problems with Pcap, we have Pcap-NG
- Pcap-NG is the “new” packet log format
- So “new” that it was introduced in ***Wireshark 1.2***
- One. Point. Two.
- 19 years ago.
- Obviously *so new* that most other tooling doesn’t use it still and it’s still worth talking about here…

Reasons to Love Pcap-NG

- Extensive metadata on packets, plus ability to add custom metadata!
- Multiple interfaces in a single capture
- Multiple link types in a single capture
- Nanosecond time
- Expandable record types
  Define your own data! Custom packet data, custom metadata

Drawbacks

- A lot of tools don't support Pcap-NG still (or don't support it fully/properly)
- Not as simple to write as traditional pcap due to a lack of simple interface libraries
- Not many tools write Pcap-NG formats
  Wireshark, Kismet … not many others?

Pcap-NG Boosterism

- I think Pcap-NG is awesome
- You might too
- Way more flexible than traditional pcap logs
- More tools should use it
- Lets talk about why

Basic Pcap-NG Internals

- Block-based log
- Basic identifier header, followed by blocks
- Blocks can define packets, capture interfaces, custom data, statistics, encryption information, etc
- Blocks can hold multiple info elements in the form of options
- Turtles all the way down

Different Block Types

- Section Header Block (SHB)
  Looks almost like a traditional PCAP header

- Interface Description Block (IDB)
  Defines a capture interface, of which there may be many

- Enhanced Packet Blocks (EPB)
  Captured packet, event, other data
  Options – arbitrary metadata - DNS resolution, statistics, comments, decryption info

Flexible Order

- Pcap-NG is a linear log – not random access
- So how do we add new content like an interface?
- Just put in another interface block before the first time we reference it!
- Need extra metadata on a packet?
- Just add extra data! Not all packets need the same metadata!

Packet Records

- Largely focused on binary packet content (but it doesn't have to be)
- Have a standard regular format
- Common packet types are 802.3, 802.11, etc. We have link types for them.
- But… what else?

Packet-Like Objects

- Don’t have an assigned link type
- Often come from low-level / raw radio captures or represent non-standardized content

Mostly Wireless

- When we say raw capture…
- I mostly mean wireless / radio
- Mainly because physically connected devices use known standards
- CAN / RS485 / SPI / Ethernet / USB / etc
- *Tons* of random RF devices with no standard

Can’t See What You Can’t See

- You can’t inspect what you can’t see
- Many consumer devices have been invisible to tools like Wireshark because there was no way to capture the packets
- “Some interference on Wi-Fi” might be all we’d see
- Now cheap SDR lets us start looking (and logging) them!

Why No Link Type?

- Often for devices captured like this there is no standard to help decode
- No strict packet format
- Often not even designed to be decoded by a PC
- We just happen to know how to decode instances of them

Example PLO Emitters

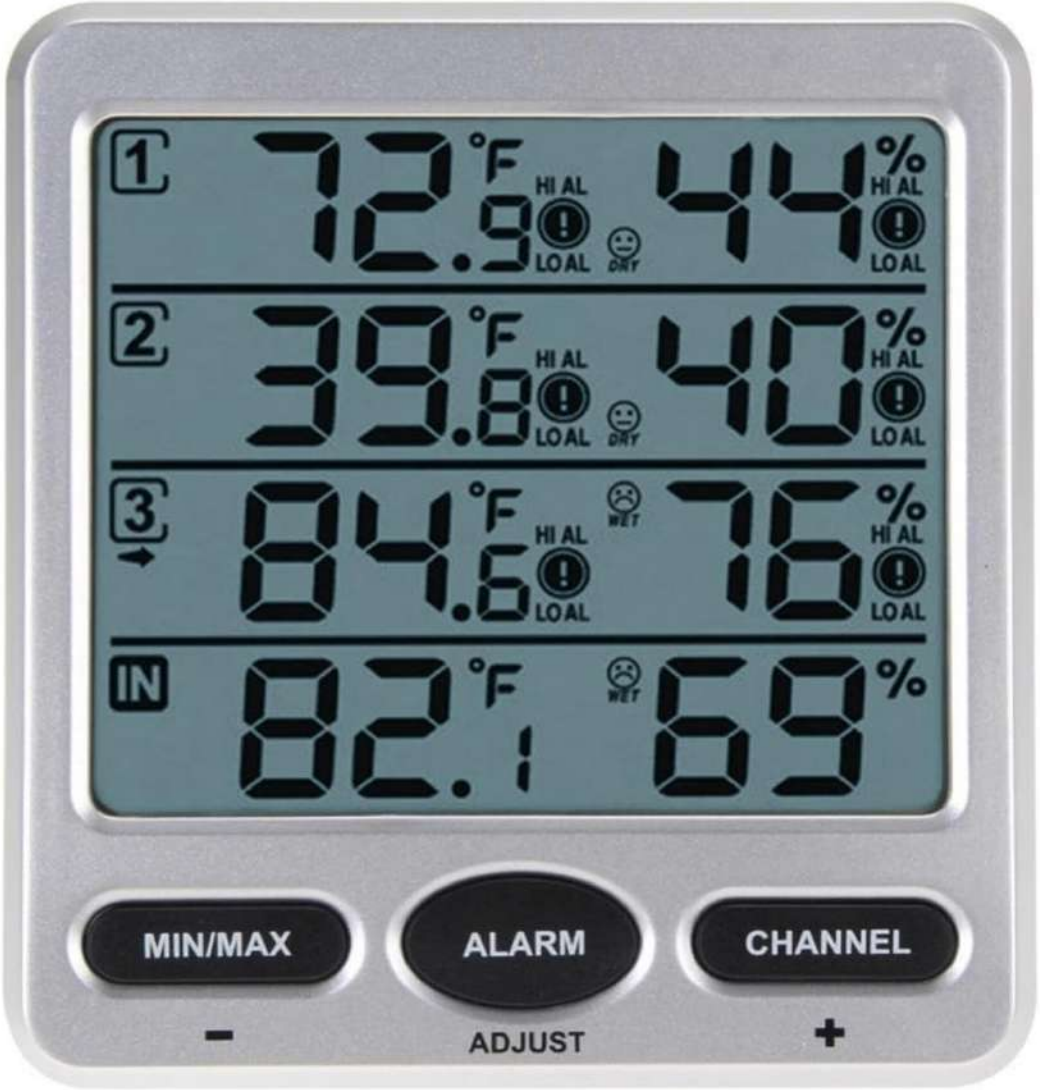
- Wireless weather stations
- Temperature sensors
- Car TPMS sensors
- Airplane, ship, etc data (one of the closest to a fixed format that could be worthy of a DLT)
- Power, water, gas meters

How do they happen?

- Packet-like data typically comes from devices which do not, or do not need to, operate on a standard

- As an example:
  - There is no standard for wireless consumer temperature sensors
  - Every company sells their own ecosystem
  - It's not Wi-Fi. It's not Bluetooth.
  - It doesn't even need to be compatible with future versions of the same product

Example Sensor



Packet-Like Enough

- Still, enough like a packet for us
- Identifiable, consistent source
- Sometimes an identifiable destination
- Comprehensible packet format

- Why is it hard to get a link type?

What is a Link Type

- Link types are basically a magic number assigned by the libpcap team
- By policy, a link type must be associated with a standard
- Once assigned, the link type fields are meant to be fixed
- This made a lot of sense when all practical packet sources come from traditional standard sources
- How do you define a DLT for a capture with no fixed standard?

No Standards

- Manufacturer isn't going to publish any standard about how it works
- It only needs to talk to a paired receiver
- Often consists of tiny amounts of data (tens of bytes at most) generated by dedicated microcontrollers

- On the other hand…
- Looks like a packet
- Quacks like a packet

Expandable by Design

- Fortunately, we can expand Pcap-NG: It was designed for it!
- Packet / block records can be customized
- Metadata can be added to existing blocks as options
- New custom metadata can be defined

Packets + Metadata

- The combination of custom packets + options let us do almost anything
- Log new packet types with a more flexible way to indicate the contents
- Log extensive metadata around traditional packet types

Packet Block Format

- Every packet block contains:
  Type
  Length
  Interface
  Timestamp
  Data
  Extra flags
- Any number of options (metadata)

Custom Block

- One of the block types is “custom”
- This allows us to add anything we need
- We just have to play by the Pcap-NG rules…
- How can anyone add custom data without interfering with someone else’s custom data?

Custom Option

- Looks a lot like a custom block (funny, that)
- Option code (ie, “Custom”)
- Length
- Content

Anti-Collision: PEN

- Among other duties (DNS root, IP assignment, timezones…) the IANA will assign a PEN
- Private Enterprise Number
- Globally unique identifier
- Free to get
- Pcap-NG uses a PEN to indicate the owner of a custom data object
- A PEN is just a simple number
- For example, the Kismet PEN is 55922

Custom Block

- Standard enhanced block
  Type (0xBAD = Custom)
  Length
  PEN number
  Arbitrary custom data
- Options – both standard and custom!

Poison PEN

- Not everything is perfect, unfortunately
- The PEN is a *top-level identifier* which indicates the custom ID of a block or option
- That gives *one* block and *one* option per PEN
- You'll probably want to be able to use multiple types in your organization.
- This means *we* need to design the data payload to allow for this!

Ways to Encode Data

Multiple ways to encode our data

1. Just shove it all into something like JSON

2. Binary encoded structures

JSON

- JSON is easy to read
- … But expensive to serialize and parse in quantity or at speed
- Not best suited for content which is primarily binary data
- Other possible strange encoding edge cases
- Reasonable for content that may already be in JSON, or which is intermittent

Possible JSON Content

- Very simple

{“GPS”: {“lat”: 123.4567, “lon”: 45.6789}}

{“device”: “rtl433_thermometer”, ”temp_c”: 30}

Binary Formats

- Already used heavily (they're what Pcap-NG is, after all)
- Efficient and fast
- Pcap-NG itself is a great example of how to structure things

Reasonable Binary Layout

- Magic signature to indicate block type
- Version
- Data length
- Field presence bitmask
- Content

Magic Signature

- Arbitrary byte value to indicate what type of sub-record this is
- Also used to validate that the sub-record is something we expect
- Similar values used in Pcap and Pcap-NG

Versioning

- You can’t go back in time and add fields you forgot
- Even if you only think you need a single version… spend two bytes for a version field!
- Allows for flexibility going forwards

Field Presence Bitmask

- Record style used by Pcap-NG, Radiotap, and others
- Simple bitfield indicates what fields are present in this record
- Allows compressed data when only some fields are needed
- Allows for adding fields in the future
- Doesn’t allow for *replacing* fields easily, plan accordingly!

Binary Data Content

- Again, our best reference is how Pcap-NG defines records
- Records can be designed as packed (unaligned) or aligned structs of data… *document which you pick*!
- Alignment with optional fields can be tricky for reading, but is simple for writing
- Alignment helps the speed of processing data

Alignment

- When in doubt, use aligned structs
- Binary data is generally aligned to word boundaries
- Typically aligned to 32 bit / 4-byte word boundaries
- Used in Pcap, Pcap-NG, most other protocols

Writing Alignment

- Easier to write aligned structs, even with dynamic fields
- Writer knows what fields it uses and can define the data structure accordingly, pre-aligned
- Asymmetric effort to read with proper alignment
- This is OK - writing needs to be closer to real-time to avoid dropping packets so optimize for that

Endian

- Multibyte values need to have a defined endian order
- Pcap-NG defines endian based on the signature magic in the file header
- Best practice is to follow the endian order of the file
- For writing speed, Pcap-NG uses the endian of the device writing the file, which is good advice

Floating Without a Paddle

- Floating point data presents a problem
- There is no standard for endian conversion of floating point data
- Safest method is to convert them to fixed precision values of sufficient fidelity

Float to Fixed

- A balance between range and precision
- At the most basic:

  Pick a large number and multiply your floating point value to get a fixed value.

  Divide by the same large number to get a floating point value on the other side.
- 123.4567 * 10000 = 1234567 as fixed point

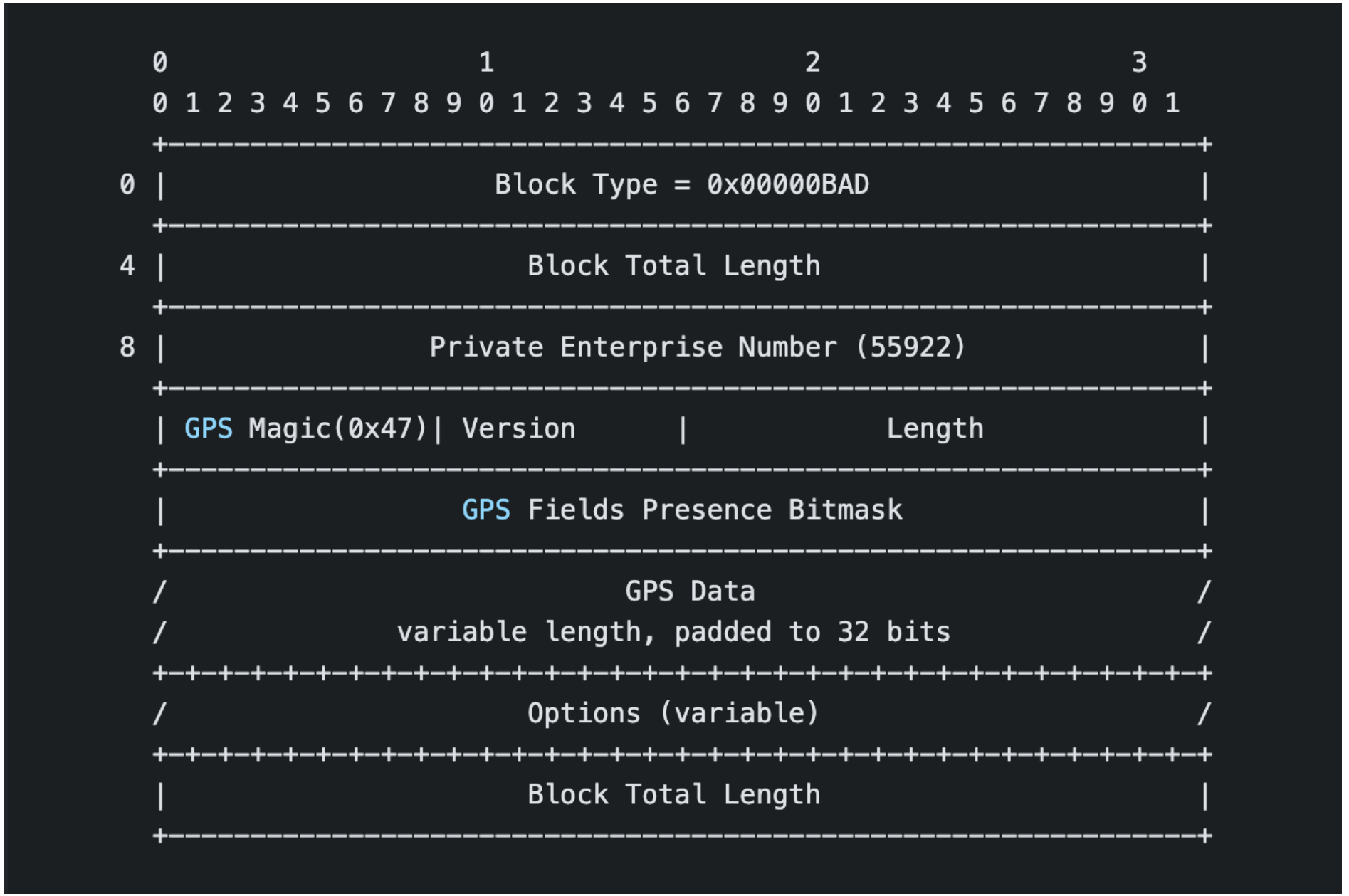
Picking Precision

- Pick your floating point multiplier appropriately
- It “doesn’t matter” as long as it’s documented
- Tune based on your requirements
- Larger multipliers yield higher precision, but lower maximum values

Kismet Geolocation Data

- As an example…
- The Pcap-NG standard was unable to settle on geolocation data
- Kismet uses a custom option to put location data derived from the PPI standard
- Fields indicated by bitfield, floats converted to fixed, etc

Putting it Together



Document!

- Ultimately whatever you decide to use will likely be fine
- *As long as you document it accurately*
- Accurate documentation of your custom options is the only way they'll be useful!

Tools For Writing Pcap-NG

- Currently no universal libpcap equivalent library
- Kismet implements a custom Pcap-NG logger
- Most other tools also implement custom loggers
- Programmatic access with Python Scapy library

Writing is Less Generic

- Being able to add extra metadata makes the packet writer a little less generic
- Anything that doesn't come from the packet feed itself requires hardware to sense it
- And code to read it
- And code to integrate it into the pcap

Reading Pcap-NG

- Again, unfortunately, no quite universal solutions
- Any tool with libPcap can read Pcap-NG if it's a single link type and single source
- Wireshark, Tshark use Pcap-NG as first-class capture format
- Support for custom Pcap-NG types is growing in Wireshark and other tools

Pcap-NG Data Fetching

- Wireshark adding more internal Pcap-NG custom option support
- This will make submitting patches for custom processing *much easier*
- Pcap-NG standard options exposed under the *frame.** filters
- Larger vendors like Apple starting to adopt Pcap-NG

Random Comments - Processing Speed

- We referenced the speed of processing several times
- For small captures, it doesn't matter
- For large captures, it can be *significant*, plan accordingly!
- Write speed is often more important than read!
- It's OK if it takes longer to process log
- It's *not OK* to lose packets because it took too long to write!
- It's entirely plausible to hit ~200,000 packets per second!

Feedback

#sf25us

- https://www.ietf.org/archive/id/draft-tuexen-opsawg-pcapng-03.html#name-enhanced-packet-block

  https://kismetwireless.net/docs/dev/pcapng_gps/