THE STATE OF HAMMER: READY FOR ICS

Sven M. Hallberg Adam Crain Meredith L. Patterson Sergey Bratus

26 May 2016
1464.2 megaseconds since the Unix epoch

LangSec workshop
at IEEE Security & Privacy 2016

RECAP: THE HAMMER PARSER COMBINATOR LIBRARY

- Primitive parsers

```
HParser *seqno = h_bits(4, false);
HParser *bit   = h_bits(1, false);
...
```

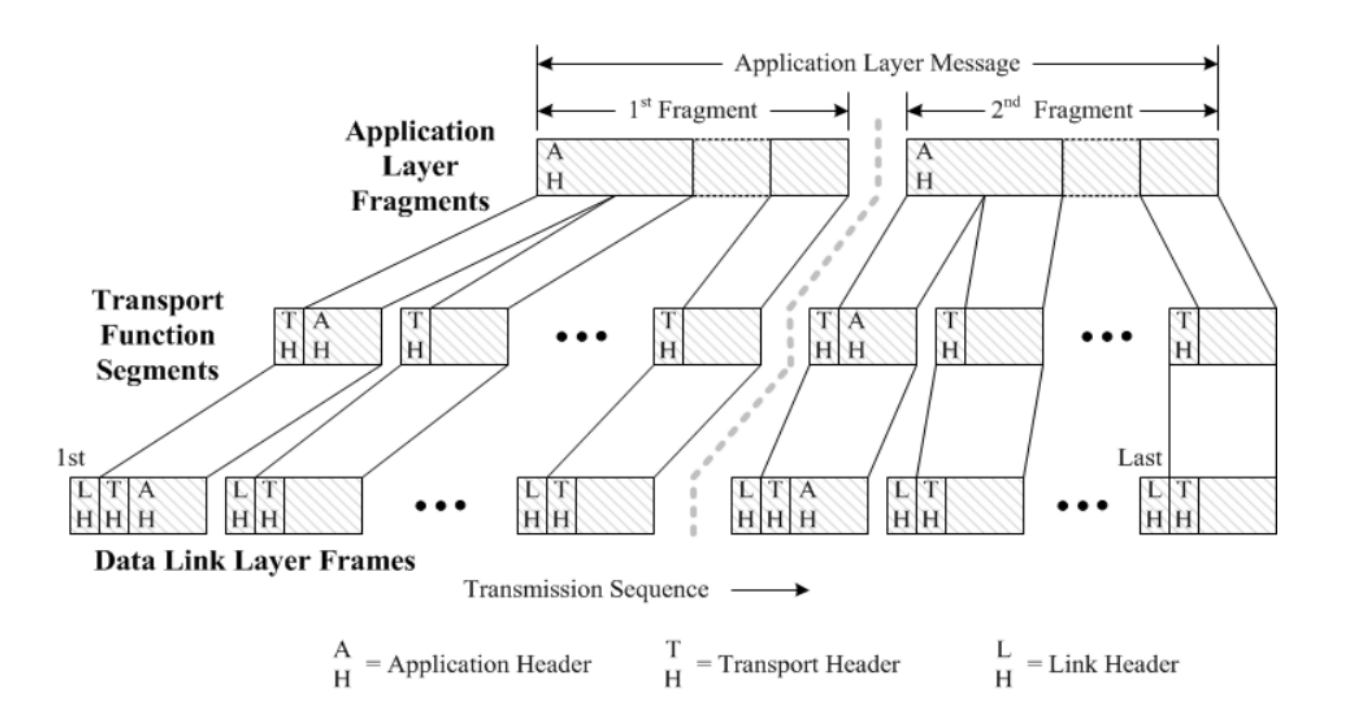
- Combined to form higher-level structures
  - `h_choice`, `h_many`, `h_many1`, ...
  - define own combinators

SPOILER

Start with a grammar, stay with a grammar!

But what is it?

DNP3 PROTOCOL LAYERS



AppHdr (ObjHdr Object*)*

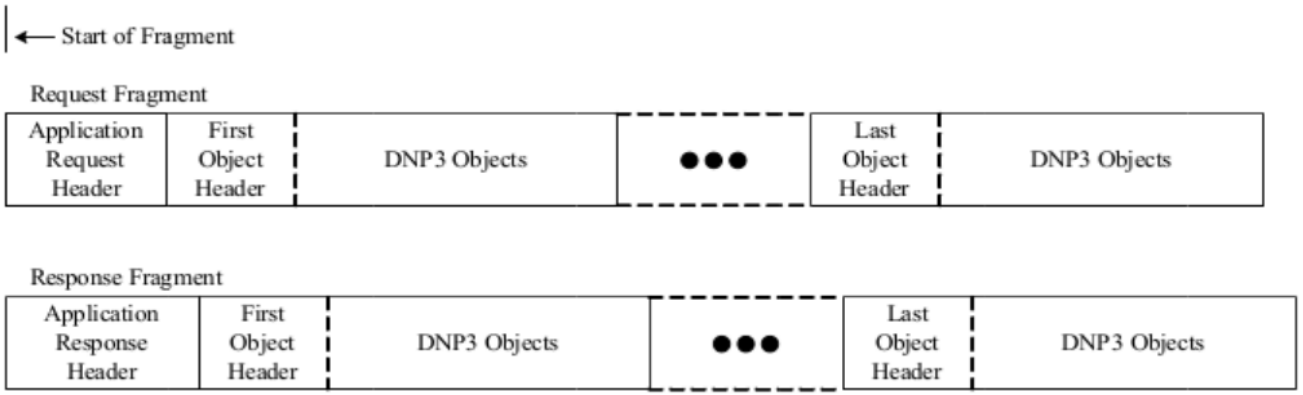


Figure 4-4—Fragment structure

```
AppHdr = SeqNo Flags FunctionCode
ObjHdr = Group Variation PC RSC Range
Object = Prefix? ObjectData
```

PROBLEM?

The previous fails to express dependencies between

- `Flags` and `FunctionCode`
- `FunctionCode` and (`Group`, `Variation`)
- `RSC`, `Range`, `PC`
- (`Group`, `Variation`) and `Objects`

DNP3 Complex!?!

Table 14-4—Level 3 implementation (DNP3-L3)

DNP3 OBJECT GROUP & VARIATION			REQUEST Master may issue Outstation shall parse		RESPONSE Master shall parse Outstation may issue	
Group num	Var num	Description	Function codes (dec)	Qualifier codes (hex)	Function codes (dec)	Qualifier codes (hex)
1	0	Binary Input—Any Variation	1 (read) 22 (assign class)	00, 01 (start-stop) 06 (no range, or all)		
1	1	Binary Input—Packed format	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
1	2	Binary Input—With flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
2	0	Binary Input Event—Any Variation	1 (read)	06 (no range, or all) 07, 08 (limited qty)		
2	1	Binary Input Event—Without time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
2	2	Binary Input Event—With absolute time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
2	3	Binary Input Event—With relative time	1 (read)	06 (no range, or all) 07, 08 (limited qty)	129 (response) 130 (unsol. resp)	17, 28 (index)
10	0	Binary Output—Any Variation	1 (read)	00, 01 (start-stop) 06 (no range, or all)		
10	2	Binary Output—Output status with flags	1 (read)	00, 01 (start-stop) 06 (no range, or all)	129 (response)	00, 01 (start-stop)
12	1	Binary Command—Control relay output block (CROB)	3 (select) 4 (operate) 5 (direct op)	17, 28 (index)	129 (response)	echo of request
			6 (dir. op, no ack)	17, 28 (index)		

GRP	VAR	Type	Description	Si
0 (0x00)	246 (0xF6)	Attribute	Device Attributes - User-assigned ID code/number	

Table 12-4—g3 double-bit binary input static objects

Group	Variation	Subset levels				Request (outstation must parse)		Response (master shall parse)	
		1	2	3	4	Function codes (decimal)	Qualifier codes (hexadecimal)	Function codes (decimal)	Qualifier codes (hexadecimal)
3	0	x	x	x		—	—		
3	0				✓	1 (READ)	00, 01, 06		
3	0				✓	22 (ASSIGN_CLASS)	00, 01, 06		
3	1	x	x	x		—	—	—	—
3	1				✓	1 (READ)	00, 01, 06	129 (RESPONSE)	00, 01
3	2	x	x	x		—	—	—	—
3	2				✓	1 (READ)	00, 01, 06	129 (RESPONSE)	00, 01

2 (0x02)	1 (0x01)	Event	Binary Input Event	1 o		
2 (0x02)	2 (0x02)	Event	Binary Input Event - with Absolute Time	7 o		
2 (0x02)	3 (0x03)	Event	Binary Input Event - with Relative Time	3 o		
3 (0x03)	0 (0x00)	Static	Double-bit Binary Input - Any Variations			
3 (0x03)	1 (0x01)	Static	Double-bit Binary Input - Packed Format	2		
3 (0x03)	2 (0x02)	Static	Double-bit Binary Input - Status with Flags	1 o		
4 (0x04)	0 (0x00)	Event	Double-bit Binary Input Event - Any Variations			
4 (0x04)	1 (0x01)	Event	Double-bit Binary Input Event	1 o		
4 (0x04)	2 (0x02)	Event	Double-bit Binary Input Event with Absolute Time	7 o		
4 (0x04)	3 (0x03)	Event	Double-bit Binary Input Event with Relative Time	3 o		
10 (0x0A)	0 (0x00)	Static	Binary Output - Any Variations		1, 22	
10 (0x0A)	1 (0x01)	Static	Binary Output - Packed Format	1 bit	1, 2	129
10 (0x0A)	2 (0x02)	Static	Binary Output - Status with Flags	1 octet	1	129
11 (0x0B)	0 (0x00)	Event	Binary Output Event - Any Variations		1	
11 (0x0B)	1 (0x01)	Event	Binary Output Event - Status	1 octet	1	129,130

A.23.1.2.3 Notes

Read requests and responses shall use qualifier code 0x07
an outstation receives this request, it implicitly indicates
current time.

This object can be included in a write request. Write reque
value of 1 for this object. When an outstation receives t
wants to set the current time in the outstation.

COMPLEXITY SHIFTING

A.23.1.2.3 Notes

Read requests and responses shall use qualifier code 0x07 and a range field value of 1 for this object. When an outstation receives this request, it implicitly indicates that the master wants the outstation to return the current time.

This object can be included in a write request. Write requests shall use qualifier code 0x07 and a range field value of 1 for this object. When an outstation receives this request, it implicitly indicates that the master wants to set the current time in the outstation.

- Syntax spills into "semantics".

COMPLEXITY SHIFTING (CONT.)

Object group 51: common time-of-occurance

An example of an object that depends on a Time and Date Common Time-of-Occurrence object is a binary input change event with relative time, object group 2, variation 3.

The following shows how multiple Time and Date CTO objects may be included in a response when there are not enough bits in a data object to hold the relative time with respect to a single Time and Date CTO object. Each data object's time is relative to the immediately preceding Time and Date CTO. In the figure, the time in DO_{i+1} is relative to $T\&D_1$:

- Syntax spills... where?!?
- "Obvious" constraints are not in spec

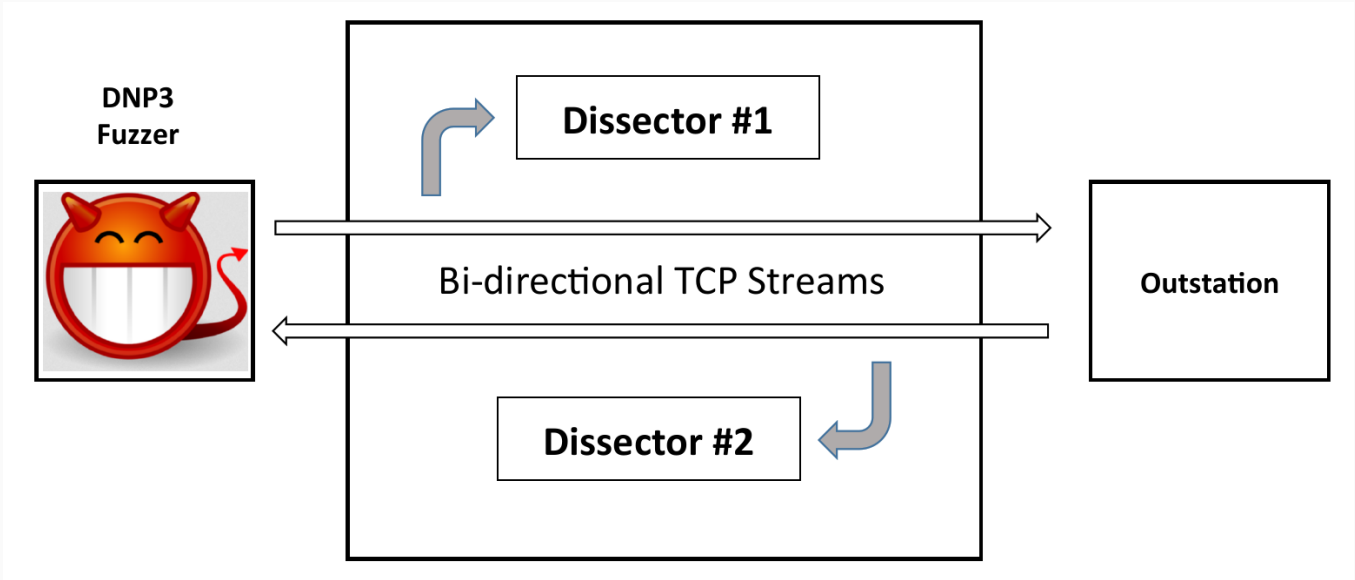
HAMMER EXAMPLE: DNP3 "SELECT" FUNCTION

```
pcb    = dnp3_p_g12v2_binoutcmd_pcb_oblock;
pcm    = dnp3_p_g12v3_binoutcmd_pcm_oblock;
crob   = dnp3_p_g12v1_binoutcmd_crob;
anaout = dnp3_p_anaout_oblock;

sel_pcb    = h_sequence(pcb, h_many1(pcm), NULL);
sel_oblock = h_choice(sel_pcb,
                      crob,
                      anaout, NULL);

select = h_many(sel_oblock);
```

APPLICATION: VALIDATING PROXY



ELFBAC INTEGRATION

- **ELFbac**: *Intra*-process memory isolation based on ELF sections
- Can also restrict access to system calls
- "Behavioral access control" to capture programmer intent

  Only main loop may call OS

  Only OS may write to input buffer

  Only parser may read from input buffer

  Rest of application can read parse results

ELFBAC CHALLENGES

- Application must be designed for separation
- Normal `malloc` not (yet) captured
- "Overcommit" makes custom heaps feasible
  - but memory accounting can still get in the way
  - we want to allocate *address space*
- Hammer worked well with custom allocator
  - reusable

VALIDATION

- Fine-grained unit tests (*lots* of unit tests)
- Tests for common DNP3 bugs

- Valgrind
- Fuzzing: AFL and Aegis

- Different compilers

RESULTS

"I survived American Fuzzy Lop"

- AFL: resource exhaustion in Hammer ($\rightarrow$ fixed)
  - length fields...
- Directed tests: integer overflow
  - count fields...

LANGSEC AS A ROADMAP

Grammar is not everything but...
it seems to *help with everything*.

- Parsers naturally decompose for testing
- Well-factored code is easier to maintain and extend
- Decomposition helps separation of privileges

POTENTIAL APPLICATION ISSUES

- Speed impact
  - sequential tries vs. table lookups
  - CFGs are *good!*
- Memory impact
  - generic data representation blows up packed structures
  - can we separate pure validation?

IMPLEMENTATION CHALLENGES

- EDSL (Hammer) learning curve
- Difficult to debug
  - opaque stack traces
  - error messages?!?
- Framework limitations?
  - allocator support
  - sync or async I/O
  - incremental processing

EXAMPLE STACK TRACE

LESSONS

- Learn a protocol's *true* syntax!
- **Don't** shift language complexity out of parsing
- **Do** isolate/capture complexity
- Avoid lengths and counts, if possible
- Use CFG parsing algorithms for network protocols (!)

Code:

github.com/pesco/dnp3
github.com/sergeybratus/proxy

ELFbac:

elfbac.org

DNP3 PROTOCOL LAYERS

- Application:

```
AppHdr (ObjHdr Object*)*
```

- Transport (think TCP):

```
(SeqNo Flags Payload)+
```

- Link (think Ethernet):

```
Header CRC (Payload CRC)*
```

HAMMER EXAMPLE: DNP3 "CROB" OBJECT

```
h_sequence(h_bits(4, false),  // op type
           bit,               // queue flag
           bit,               // clear flag
           tcc,
           h_uint8(),         // count
           h_uint32(),        // on-time [ms]
           h_uint32(),        // off-time [ms]
           status,            // 7 bits
           dnp3_p_reserved(1),
           NULL));
```

APPHDR IN OUR PARSER (EXCERPT)

```
H_RULE (confc,  dnp3_p_int_exact(fc, DNP3_CONFIRM));
H_RULE (reqfc,  h_int_range(fc, 0x01, 0x21));
H_RULE (unsfc,  h_choice(fc_ur, fc_ar, NULL));
H_RULE (rspfc,  h_choice(fc_rsp, fc_ar, NULL));

H_RULE (anyreqfc,   h_choice(confc, reqfc, NULL));
H_RULE (anyrspfc,   h_choice(unsfc, rspfc, NULL));

H_ARULE(ereqfc,     h_right(h_and(h_not(anyreqfc)), fc));
H_ARULE(erspfc,     h_right(h_and(h_not(anyrspfc)), fc));

H_RULE (req_header, h_choice(h_sequence(conac, confc, NULL),
                             h_sequence(reqac, reqfc, NULL),
                             h_sequence(anyreqac, ereqfc, NULL), NULL));
H_RULE (rsp_header, h_choice(h_sequence(unsac, unsfc, iin, NULL),
                             h_sequence(rspac, rspfc, iin, NULL),
                             h_sequence(anyrspac, erspfc, iin, NULL), NULL));
```

DNP3 could be context-free

Obstacles:

- Packet lengths
- Object counts
- Syntactic error handling
- Complex header field relations

PARSING VARIABLE-LENGTH DATA

DNP3 link layer:

```
Header CRC (Payload CRC)*
```

- Generate parsers for CRC'ed blocks of size 1-16
- Extend to parsers for sizes 0-255
- Dispatch on length field to appropriate parser