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
  
  ```c
  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
Start with a grammar, stay with a grammar!

But what is it?
DNP3 PROTOCOL LAYERS

Application Layer Message

1st Fragment

2nd Fragment

Transport Function Segments

Data Link Layer Frames

Transmission Sequence

AH = Application Header
TH = Transport Header
LH = Link Header
AppHdr = SeqNo Flags FunctionCode
ObjHdr = Group Variation PC RSC Range
Object = Prefix? ObjectData
The previous fails to express dependencies between

- Flags and FunctionCode
- FunctionCode and (Group, Variation)
- RSC, Range, PC
- (Group, Variation) and Objects
A Peek Into the Spec

DNP3 Complex!?!?

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

<table>
<thead>
<tr>
<th>Group</th>
<th>Variation</th>
<th>Subset</th>
<th>Request (output must parse)</th>
<th>Response (master shall parse)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>1 (READ) 00, 01, 06</td>
<td>02 (RESPONSE) 00, 01</td>
</tr>
<tr>
<td>3</td>
<td>0</td>
<td>0</td>
<td>22 (ASSIGN_CLASS) 00, 01, 06</td>
<td>02 (RESPONSE) 00, 01</td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>0</td>
<td>1 (READ) 00, 01, 06</td>
<td>129 (RESPONSE) 00, 01</td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>0</td>
<td>1 (READ) 00, 01, 06</td>
<td>129 (RESPONSE) 00, 01</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>1 (READ) 00, 01, 06</td>
<td>129 (RESPONSE) 00, 01</td>
</tr>
</tbody>
</table>

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 request value of 1 for this object. When an outstation receives it wants to set the current time in the outstation.

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

<table>
<thead>
<tr>
<th>Group</th>
<th>Variation</th>
<th>Subset</th>
<th>Request (output must parse)</th>
<th>Response (master shall parse)</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>Binary Input—Any Variation 0</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>0</td>
<td>Binary Input—Packed Format 1</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>Binary Input—Write Flags 2</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>0</td>
<td>Binary Input—Write time 2</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>0</td>
<td>Binary Input—Write absolute time 2</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>3</td>
<td>2</td>
<td>0</td>
<td>Binary Input—Write relative time 2</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>Binary Output—Any Variation 0</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>Binary Output—Any Variation 1</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
<td>Binary Output—Any Variation 2</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>3</td>
<td>0</td>
<td>Binary Output—Any Variation 3</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>4</td>
<td>0</td>
<td>Binary Output—Any Variation 4</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>5</td>
<td>0</td>
<td>Binary Output—Any Variation 5</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>6</td>
<td>0</td>
<td>Binary Output—Any Variation 6</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>7</td>
<td>0</td>
<td>Binary Output—Any Variation 7</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>8</td>
<td>0</td>
<td>Binary Output—Any Variation 8</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>9</td>
<td>0</td>
<td>Binary Output—Any Variation 9</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>0</td>
<td>Binary Output—Any Variation 10</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>11</td>
<td>0</td>
<td>Binary Output—Any Variation 11</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>12</td>
<td>0</td>
<td>Binary Output—Any Variation 12</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>13</td>
<td>0</td>
<td>Binary Output—Any Variation 13</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>14</td>
<td>0</td>
<td>Binary Output—Any Variation 14</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>15</td>
<td>0</td>
<td>Binary Output—Any Variation 15</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>16</td>
<td>0</td>
<td>Binary Output—Any Variation 16</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>17</td>
<td>0</td>
<td>Binary Output—Any Variation 17</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>18</td>
<td>0</td>
<td>Binary Output—Any Variation 18</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>19</td>
<td>0</td>
<td>Binary Output—Any Variation 19</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>20</td>
<td>0</td>
<td>Binary Output—Any Variation 20</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>21</td>
<td>0</td>
<td>Binary Output—Any Variation 21</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>22</td>
<td>0</td>
<td>Binary Output—Any Variation 22</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>23</td>
<td>0</td>
<td>Binary Output—Any Variation 23</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>24</td>
<td>0</td>
<td>Binary Output—Any Variation 24</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>25</td>
<td>0</td>
<td>Binary Output—Any Variation 25</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>26</td>
<td>0</td>
<td>Binary Output—Any Variation 26</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>27</td>
<td>0</td>
<td>Binary Output—Any Variation 27</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>28</td>
<td>0</td>
<td>Binary Output—Any Variation 28</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>29</td>
<td>0</td>
<td>Binary Output—Any Variation 29</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>30</td>
<td>0</td>
<td>Binary Output—Any Variation 30</td>
<td>Read 00, 01 (status or all)</td>
</tr>
<tr>
<td>0</td>
<td>31</td>
<td>0</td>
<td>Binary Output—Any Variation 31</td>
<td>Read 00, 01 (status or all)</td>
</tr>
</tbody>
</table>

0x07 (status or all) 17, 18 (index) 129 (response) 00, 01 (status or all)
• Syntax spills into "semantics".
Object group 51: common time-of-occurrence

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
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

DNP3 Fuzzer

Dissector #1

Bi-directional TCP Streams

Dissector #2

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

• Valgrind
• Fuzzing: AFL and Aegis

• Different compilers
"I survived American Fuzzy Lop"

- AFL: resource exhaustion in Hammer (→ fixed)
  - length fields...
- Directed tests: integer overflow
  - count fields...
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
• 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
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:
  \[ \text{AppHdr (ObjHdr Object\*)*} \]

- Transport (think TCP):
  \[ (\text{SeqNo Flags Payload})* \]

- Link (think Ethernet):
  \[ \text{Header CRC (Payload CRC)*} \]
hammer example: dnp3 "crob" object

```c
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));
```
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
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