WIP: Towards a Platform to Compare Binary Parser Generators

Olivier Levillain  Sébastien Naud  Aina Toky Rasoamanana
Télécom SudParis

May 28th 2021
LangSec Workshop
Binary parsers: a problem, many solutions?

Problem

▶ writing parsers for complex specifications is hard
▶ especially when said specifications are ambiguous

Solution(s)

▶ describe the formats to parse in a simple(r) form, typically a DSL
▶ use a parser generator to transform this “description” into code
▶ many proposals with different approaches in different languages
▶ Nail (LangSec 2014)
▶ Parsifal (LangSec 2014)
▶ Nom (LangSec 2015 and 2017)
▶ Hammer (DNP3 case study at LangSec 2015)
▶ Parsley (LangSec 2020)
▶ ...
Binary parsers : a problem, many solutions?

Problem

- writing parsers for complex specifications is hard
- especially when said specifications are ambiguous

Solution(s)

- describe the formats to parse in a simple(r) form, typically a DSL
- use a parser generator to transform this “description” into code
Binary parsers: a problem, many solutions?

Problem

▶ writing parsers for complex specifications is hard
▶ especially when said specifications are ambiguous

Solution(s)

▶ describe the formats to parse in a simple(r) form, typically a DSL
▶ use a parser generator to transform this “description” into code
▶ many proposals with different approaches in different languages
  ▶ Nail (LangSec 2014)
  ▶ Parsifal (LangSec 2014)
  ▶ Nom (LangSec 2015 and 2017)
  ▶ Hammer (DNP3 case study at LangSec 2015)
  ▶ Parsley (LangSec 2020)
  ▶ ...
langsec-pf : a platform to compare tools
langsec-pf: current status

Tools

- Hammer (C, 2012)
- Kaitai Struct (various languages, 2016)
- Nail (C, 2014)
- Netzob (Python, 2011)
- Nom (Rust, 2014)
- Parsifal (OCaml, 2011)
- RecordFlux (Python/Ada, 2018)

Specs

- Basic constructions (magic number, string, list)
- ICMP, IP
- DNS
- ASN.1 / X.509
Demo
Features

The current platform has several features

▶ sample validation (to compare the implementations)
▶ fuzzing with AFL and libfuzzer (only for C/C++ programs)
▶ interactive shell to mess around
Discussion about the expressiveness

Limits to the descriptive part

- DSLs are usually pretty
- but they require external help to handle complex constructions

Comparison of various DNS implementations

<table>
<thead>
<tr>
<th>Descr.</th>
<th>Code</th>
<th>Total</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>Hammer</td>
<td>105</td>
<td>158</td>
<td>263 No compression</td>
</tr>
<tr>
<td>Kaitai Struct</td>
<td>231</td>
<td>0</td>
<td>231 Compression pointers not validated</td>
</tr>
<tr>
<td>Nail</td>
<td>39</td>
<td>70</td>
<td>109 Separate files</td>
</tr>
<tr>
<td>Parsifal</td>
<td>130</td>
<td>79</td>
<td>209 Many enums and RR interpretation</td>
</tr>
</tbody>
</table>
First results about the robustness

Bugs or interesting behaviours

- assertion failures in DNS Hammer implementation
- strange IO exception on truncated inputs in Kaitai Struct
- integer overflows in the code generated by Nail
- endless loops in DNS label decompression with Nail implementation
Conclusion and perspective

Current status

► an open framework with several parser generators
► mostly trivial specs and implementations for now
► a first lab to experiment on tool expressivity, robustness and performance
► this is very much a WIP!

Next steps

► include more tools and more specs
► invite tool developers to contribute
  ► an actual documentation of tool languages
  ► a framework to compare different approaches
Questions

?