Parsers

Stefan Boesen, Dartmouth College

Parsers

- Parsers eat data and perform computation. Parsers live in tons of stuff:
- web browsers
- executable loaders
- compilers
- scripting language interpreters

Parsers

- Eat data specified as a grammar
- Grammar is a set of rules that make up a language
- Parser should enforce those rules
- Only accept input that matches

On grammars…

- Often grammars are not specified.
- There's still rules a parser must enforce.
- Those conditions are usually handwritten and scattered throughout the code.
- Those conditions still make up a grammar, the grammar is just implicit.

Motivation

x509

- x509 is the specification for SSL/TLS certificates
- It's what gives you some level of security you're talking to paypal.com or amazon.com
- In 2010, researchers found that the specification was vague in handling of NULL bytes
- Left up to the implementer

x509

- Turns out that's not so great - could get a certificate for "www.paypal.com\0www.badsite.com"
- Browser would stop reading at the \0 byte
- Certificate Authority (who signs certificates) would recognize that it is a totally different website from paypal.com and issue the cert.

Strings

- Turns out the strings program parses ELF headers if you give it an ELF file

```
while (--n_elt != 0)
  if ((++idx)->shdr->bfd_section)
    elf_sec_group (idx->shdr->bfd_section) = shdr->bfd_section;
```

- One mistake and it resulted in memory corruption

Android signature verification

- Android packages (APKs) are zip archives, and contain signatures

- 1st parser unpacks and verifies signature of first file in zip with given filename

- 2nd parser runs second file in zip with given filename

- Easy signature bypass

Word RTF

- This week Microsoft just released a patch to Word's parsing of RTF files.
- ACE
- I looked: .rtf spec - 500 pages.
- Search for "rtf" turns up 59 CVEs, two this year
- .docx OfficeOpen spec - 5000+ pages

Path to a Solution

- Parsing code often is scattered around the code.
- This makes it very hard to maintain.
- This makes it very hard to audit.

A new hope

- Syntax checking should be distinct from semantic actions

- Recognizer and Processor

Sanitization

- Sanitization is blacklisting.
- PHP's magic_quotes is blacklisting.
- "Imagine everything that isn't an elephant"

Language Classes

- Regular, context-free, context-sensitive, recursively enumerable

- The more context sensitive the language the more complicated it is to parse.

- Offsets, back references, length fields all add complexity to a language

Undecidable

- It's undecidable if one CF (or worse) grammar produces the same language as another CF grammar
- There is no general purpose algorithm
- Should be as clear as possible exactly what you are parsing
- Hammer code is easier to audit

Hammer Bleed

- Heartbleed resulted from not checking length matched input
- h_length_value(h_uint16(), payload_byte)
- Creates a recognizer that only accepts input where the rule is true
- Allows us to attach semantic actions cleanly

Keep in mind

- There's no silver bullet
- Mistakes can happen in semantic actions as well
- We can at least make it easier to audit what a parser expects

Keep in mind

- Hammer is a work in progress
- Some things can be tricky to parse
- That includes our very own DNP3
- Doing things with while() and if() isn’t going to cut it

Questions?

- stefan@cs.dartmouth.edu

- https://github.com/UpstandingHackers/hammer