Nail

A Practical Tool for Parsing and Generating Data Formats

Julian Bangert, Nickolai Zeldovich
MIT CSAIL

OSDI '14
October 2014

Motivation

Parsing Vulnerabilities

- hand-crafted input parsing and output generation
- memory corruption and logic errors
- exploitable errors: Evasi0n jailbreak, X.509 parsers, Android Master Key bug

Motivation

Parsing Vulnerabilities

- hand-crafted input parsing and output generation
- memory corruption and logic errors
- exploitable errors: Evasi0n jailbreak, X.509 parsers, Android Master Key bug

Classification	Example CVE	Example description	Count
Memory corruption	CVE-2013-5660	Buffer overflow	11
Parsing inconsistency	CVE-2013-1462	Multiple virus scanners interpret ZIP files incorrectly	4
Semantic misunderstanding	CVE-2014-2319	Weak cryptography used even if user selects AES	1
Total of all vulnerabilities related to .zip processing			**16**

Motivation

Parsing Vulnerabilities

- hand-crafted input parsing and output generation
- memory corruption and logic errors
- exploitable errors: Evasi0n jailbreak, X.509 parsers, Android Master Key bug

Causes

- *semantic actions* to update application state from AST
- output generation separated from input parsing
- redundancies and dependencies in data formats

Design

- grammar
  - $\Rightarrow$ type declaration for internal model
  - $\Rightarrow$ parser (external format $\rightarrow$ internal model)
  - $\Rightarrow$ generator (external format $\leftarrow$ internal model)
- *semantic bijection* due to discarded *constants*
- *dependent fields*, e.g. lengths, checksums, offsets
- *transformations* to hold arbitrary code

Syntax I

Nail grammar	External format	Internal data type in C
`uint4`	4-bit unsigned integer	`uint8_t`
`int32 | [1,5..255,512]`	Signed 32-bit integer $x \in \{1, 5..255, 512\}$	`int32_t`
`uint8 = 0`	8-bit constant with value 0	`/* empty */`
`optional int8 | 16..`	8-bit integer ≥ 16 or nothing	`int8_t *`

`many int8 | ![0]`	A NULL-terminated string	`struct {`
		`  size_t N_count;`
		`  int_t *elem;`
		`};`

`{`
`  hours uint8`
`  minutes uint8`
`}`
	Structure with two fields	`struct {`
		`  uint8_t hours;`
		`  uint8_t minutes;`
		`};`

`<int8='"'; p; int8='"'>`	A value described by parser p, in quotes	The data type of p

Syntax II

`choose {`
`  A = uint8 | 1..8`
`  B = uint16 | 256..`
`}`
	Either an 8-bit integer between 1 and 8, or a 16-bit integer larger than 256	`struct {`
		`  enum {A, B} N_type;`
		`  union {`
		`  uint8_t a;`
		`  uint16_t b;`
		`  };`
		`};`

`@valuelen uint16`
`value n_of @valuelen uint8`
	A 16-bit length field, followed by that many bytes	`struct {`
		`  size_t N_count;`
		`  uint8_t *elem;`
		`};`

`$data transform` `  deflate($current @method)`	Applies programmer-specified function to create new stream (§4.4)	`/* empty */`
`apply $stream` *p*	Apply parser *p* to stream `$stream` (§4.4)	The data type of *p*
`foo =` *p*	Define rule `foo` as parser *p*	`typedef /* type of p */ foo;`
`*` p	Apply parser *p*	Pointer to the data type of *p*

Example – Sums and Products of Integers

```
expr = choose {
  PAREN = <uint8='('; *expr; uint8=')'>
  PRODUCT = sepBy1 uint8='*' expr
  SUM = sepBy1 uint8='+' expr
  INTEGER = many1 uint8 | '0' .. '9'
}
```

Implementation

Prototype

- supporting C (C++ in development)
- parses Nail grammars with Nail
- 130 lines grammar, 2000 lines C++
- `https://github.com/jbangert/nail/`

Implementation

Prototype

- supporting C (C++ in development)
- parses Nail grammars with Nail
- 130 lines grammar, 2000 lines C++
- `https://github.com/jbangert/nail/`

Hardening

- *arenas*
  - large, fixed-size memory allocations
  - zeroed and freed as a whole
  - one used during parsing, one for internal data
- input zeroed after successful parse
- fail-fast to avoid “fixing” malformed input

Evaluation – Data Formats

Protocol	**LoC**	**Challenging features**
DNS packets	48+64	Label compression, count fields
ZIP archives	92+78	Checksums, offsets, variable length trailer, compression
Ethernet	16+0	—
ARP	10+0	—
IP	25+0	Total length field, options
UDP	7+0	Checksum, length field
ICMP	5+0	Checksum

Evaluation – Programmer Effort

DNS server

- parse zone file, listen for requests, respond
- 183 lines C + 48 lines grammar + 64 lines C for transformations
- Hammer toy DNS: 683 lines C + 52 lines grammar

ZIP file extractor

- DEFLATE decompression of files from ZIP archive
- 50 lines C + 92 lines grammar + 78 lines C for transformations
- `extract.c` from Info-Zip unzip: 1,600 lines C

Evaluation – Security

DNS server

- no crash or heap/stack corruption during 4 hour run of Metasploit DNS fuzzer

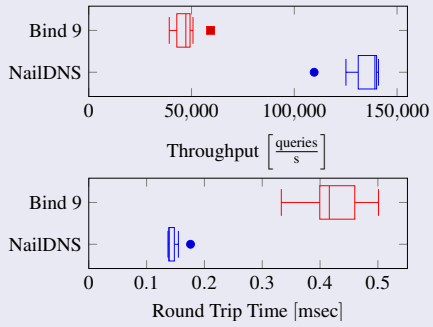
ZIP file extractor

- no memory corruptions by design (offset checks, no exposure of untrusted pointers)
- explicit encoding of redundant information
- grammar reusable for other applications

Evaluation – Performance

DNS server

- compare to ISC BIND 9 release 9.9.5
- Intel i7-3610QM, 12 GiB RAM



ZIP file extractor

???

Summary

- grammar $\rightarrow$ internal model $+$ parser $+$ generator
- avoid memory corruption and inconsistencies
- suitable for real-world (binary) formats

http://ruthe.de/cartoon/2806/datum/asc/