Automatically Fuzzing Routing Protocols

Martin Vivian, Olivier Bonaventure

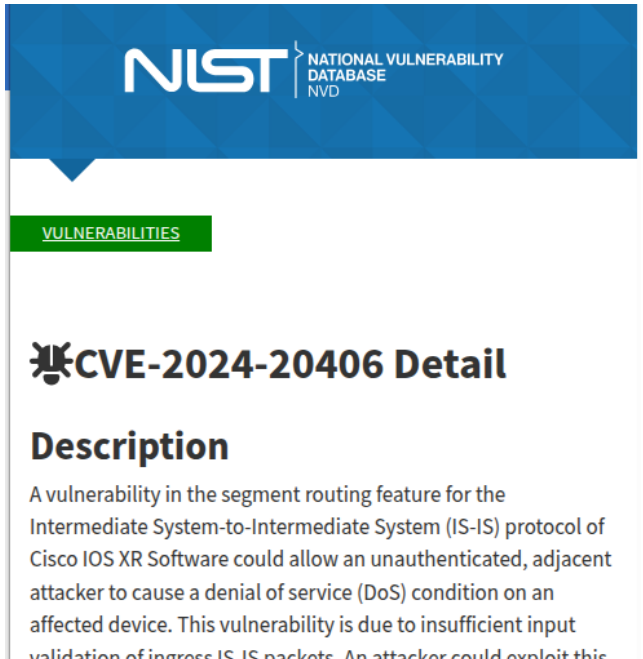
IFIP 2026, Lugano 26 May

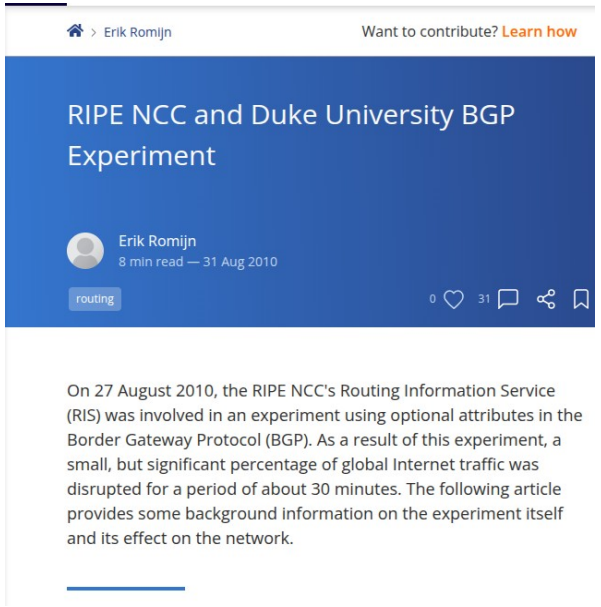
This work was supported by the Walloon Region of Belgium under the conventions n°2110186 (Cyber Excellence)

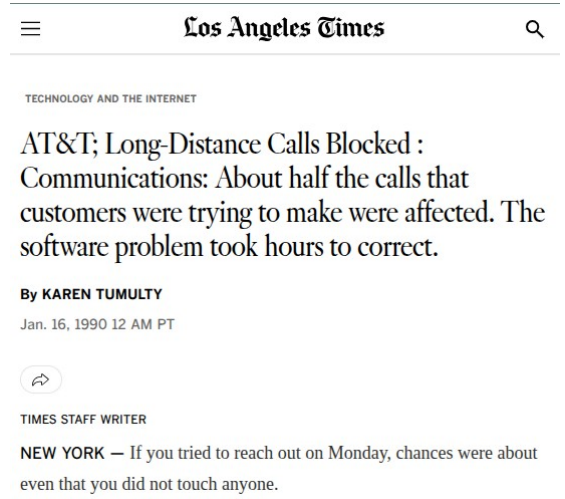
UCLouvain

Why Test Routing Protocols?

- Critical infrastructure
- Crashable via malformed messages
- Proven by real incidents







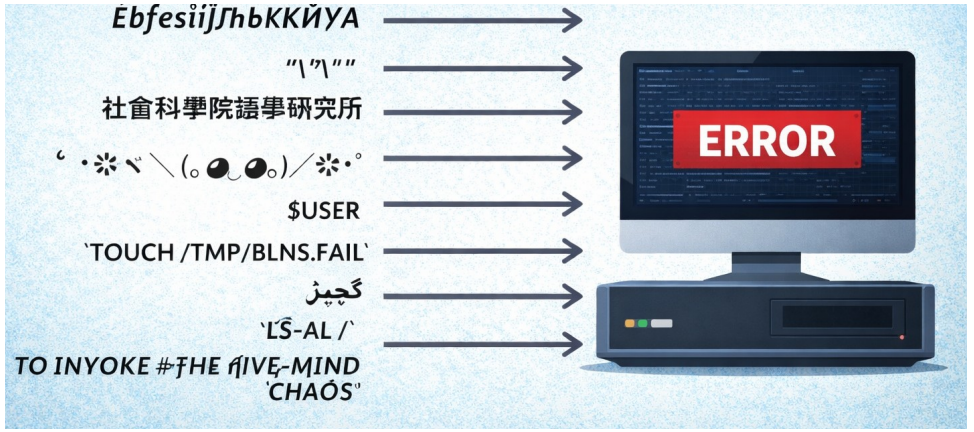
Problem

- Network communication
- Proprietary
- Non-standard format
- Checksum

Can we test routing protocols without knowing their format?

What is fuzzing ?

Automated testing with pseudo-random inputs to find crashes, bugs



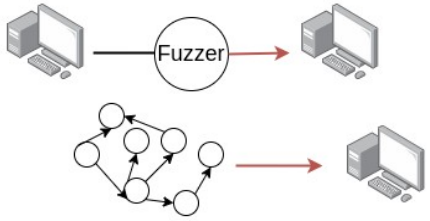
Protocols

- **Fuzzing approaches:** Black-box, Grey-box, White-box
- **Typical challenges:** proprietary formats, complex structure, high message rate, limited visibility into implementations

Fuzzing state of the art

Protocol fuzzing

- MitM
- Stateful



Unsupported by existing tools

- Non-text-based messages
- No clear request-response sequences
- Specifics for one implementation

We propose two tools

PROtocol Structure Extractor (PROSE)

- Checksum
- Header Length parameter
- Type and Length from TLV

Mutation & Stress Engine (MUSE)

- Fuzzing tool
- Generate new input to the target
- Priorities mutation on value field

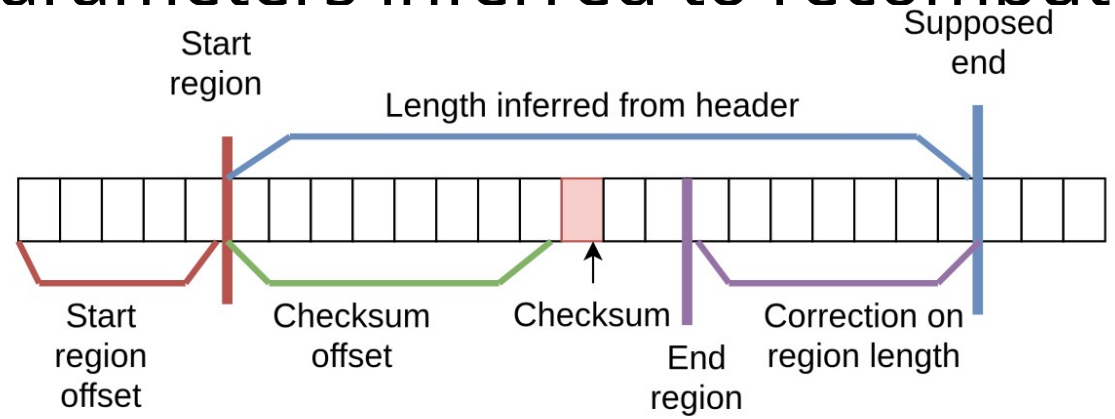
=> PROSE infers a structure reused by MUSE for fuzzing

PROSE

- Locate length field
- Inferring the parameters for checksum
- Detecting where TLV structures begin
- Automatic exploration of different hypotheses to find the Type and Length sizes
- Return a set of Type-Length

Header Length field & Checksum inference

- **BinaryInferno[*]** only helps detect the header length field
  => Avoid misinterpretation
- Invalid checksums cause immediate rejection
- Test common checksums (Fletcher-16, Internet checksum)
- Reuse inferred length in checksum computation
- 3 parameters inferred to recompute checksum



[*]Chandler, J., Wick, A., & Fisher, K. (2023, March). BinaryInferno: A Semantic-Driven Approach to Field Inference for Binary Message Formats. In *NDSS*.

PROSE: Hypothesis on TLV structure

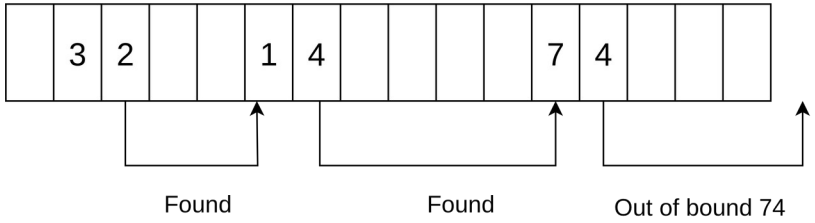
- Type followed by Length then Value
- Fixed sizes for T and L
- No padding
- Length interpreted as several bytes

PROSE: Algorithm to find the TLV

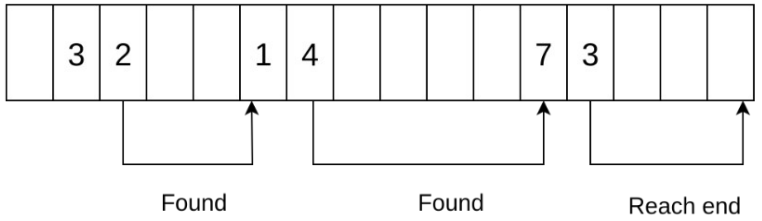
- Infer the first TLV to discover the others
- Jump through packets using inferred sizes

Exemple

1. The algorithm tests TL 32
2. After 2, it tests TL 14
3. After 4, it tests TL 74
4. After 4, it is out of bound



1. The algorithm tests TL 32
2. After 2, it tests TL14
3. After 4, it tests TL 73
4. After 3, it reach TL end



- Select the candidate position that works the most
- Infer remaining TLVs through successive jumps

Sub-TLV inference

Challenge

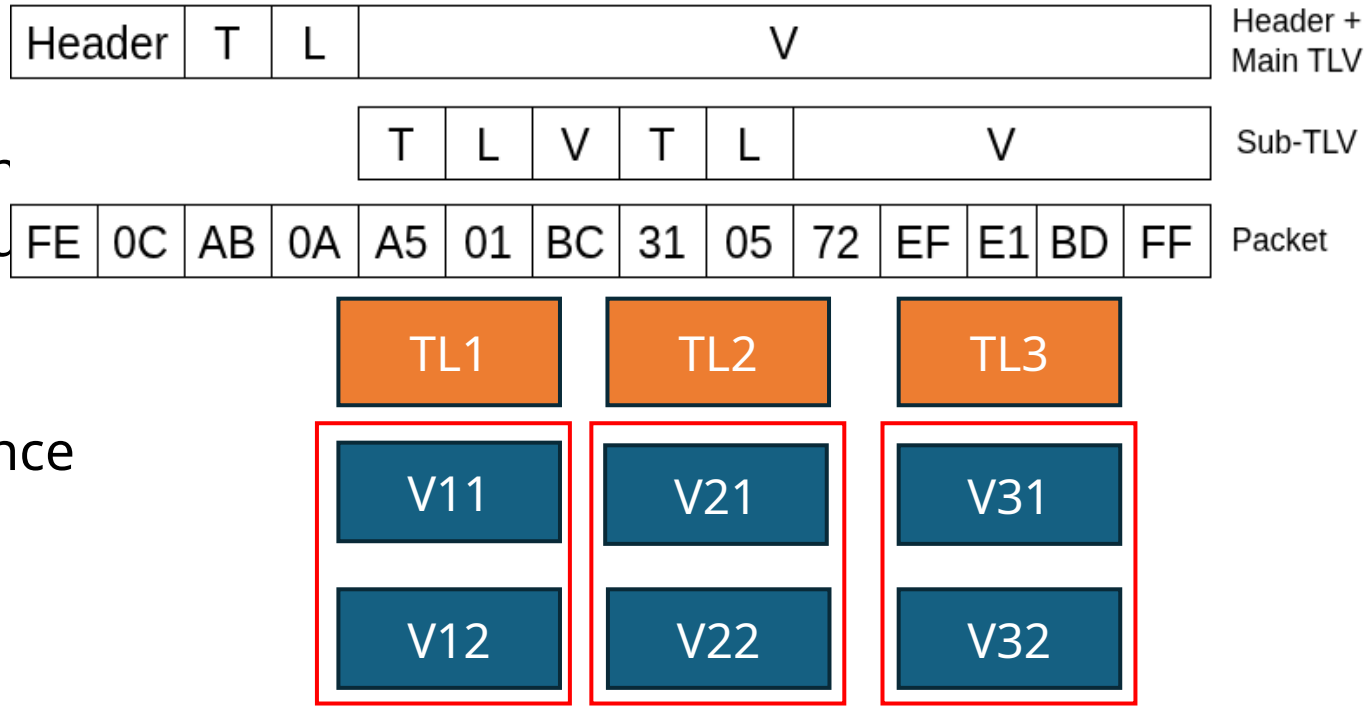
- Jump inference misses nested TLV structures
- Same length does not imply same internal structure
- Sub-TLV may start at different offsets

Idea

Messages sharing the same TL patterr
are likely to contain similar sub-structu

Approach

- Group packets by inferred TL sequence
- Extract corresponding Value fields
- Re-run inference inside each cluster



Type and Length size inference

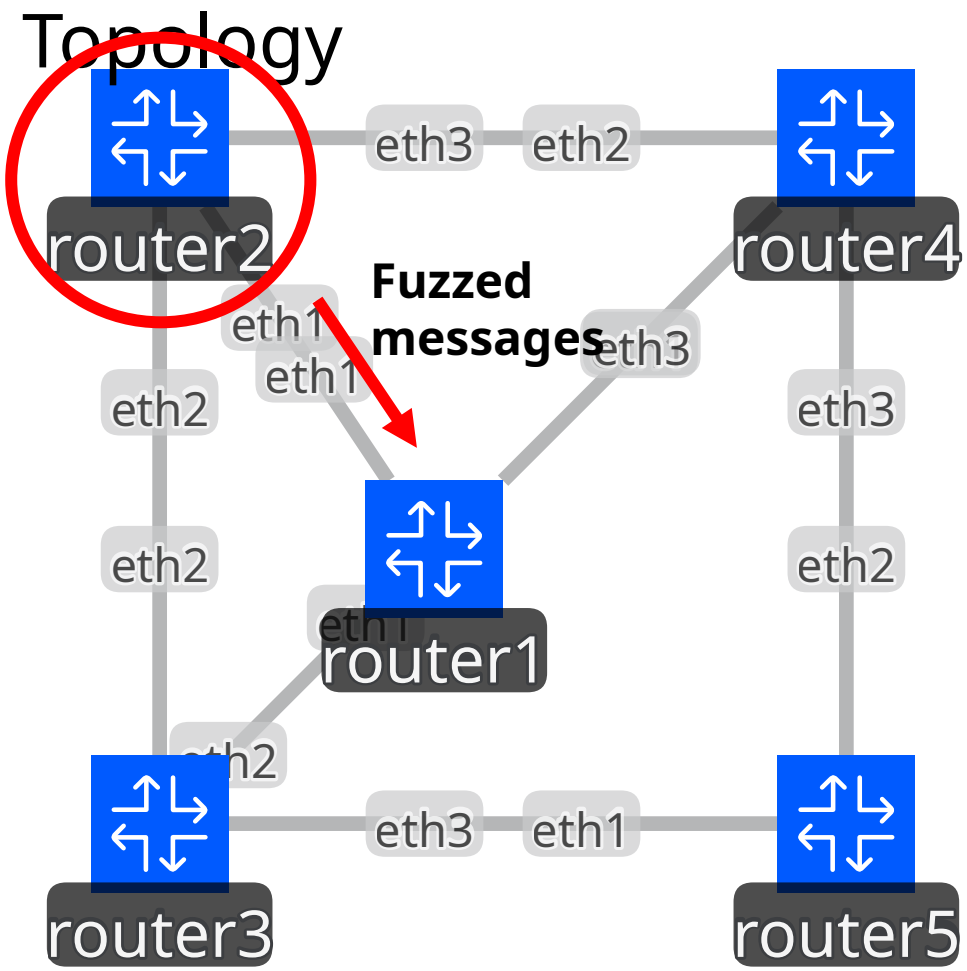
Problem

- Type and Length sizes are unknown
- Length decoding depends on assumed field sizes

Solution

- Re-run the algorithm with different Type/Length size assumptions
- Evaluate each configuration
- Keep the one producing the largest set of TL pairs
- Default maximum size: 5 bytes

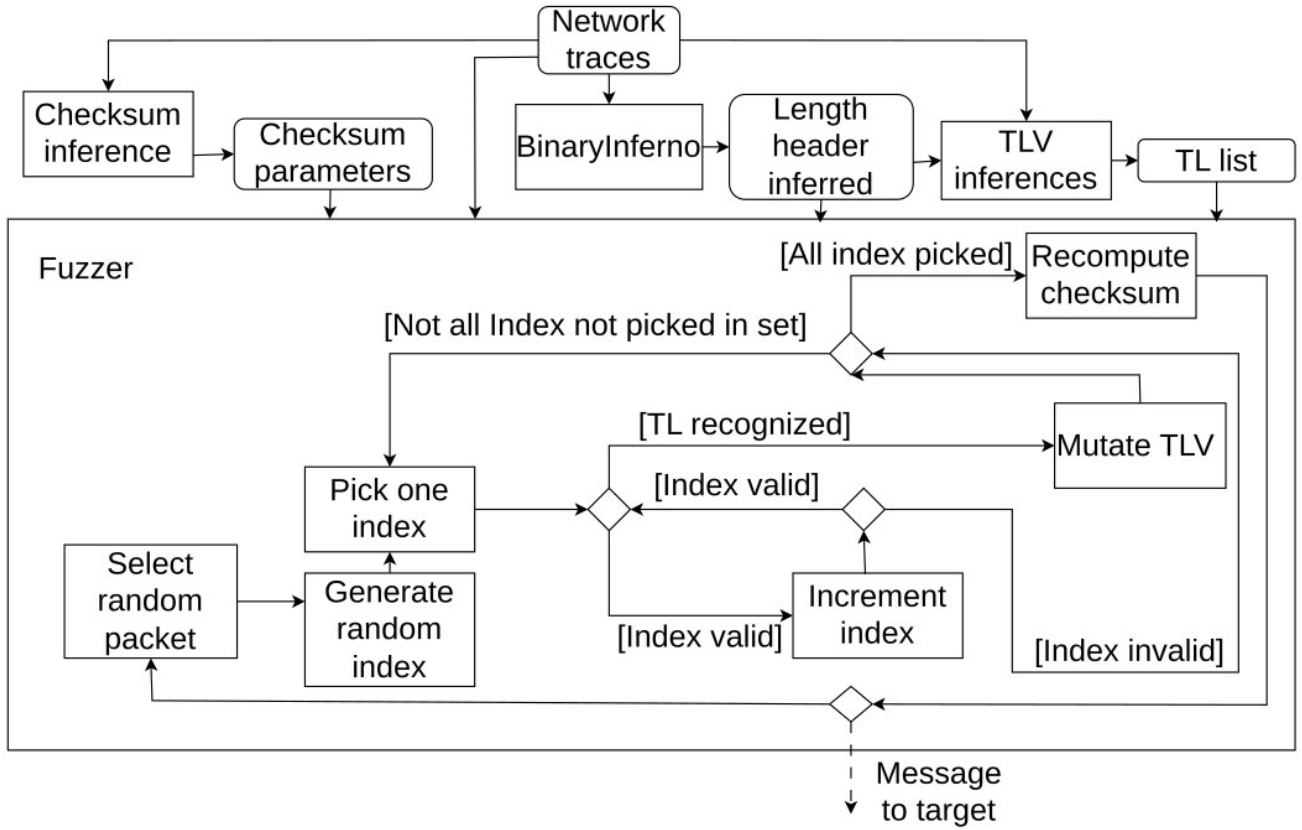
Fuzzing on routing protocols with MUSE



- MUSE runs on router2
- Constructs messages from PCAP traces
- Sends messages to router1 using inferred TLV structure (PROSE)

How messages are constructed

- MUSE receives parameters inferred by PROSE (checksum, length header, TL list)
- Select a random packet from the inference dataset
- Find TL structure in the packet
- Mutate Value field (and optionally Type/Length)
- Recompute dependent fields (Length, checksum)
- Send mutated packet



Targets for fuzzing

FRRouting 10.3

EIGRP (Enhanced Interior Gateway Routing Protocol)

- RFC 7868
- Developed by Cisco
- Proprietary until 2013
- Distance Vector

Babel

- RFC 8966
- Distance Vector
- More recent

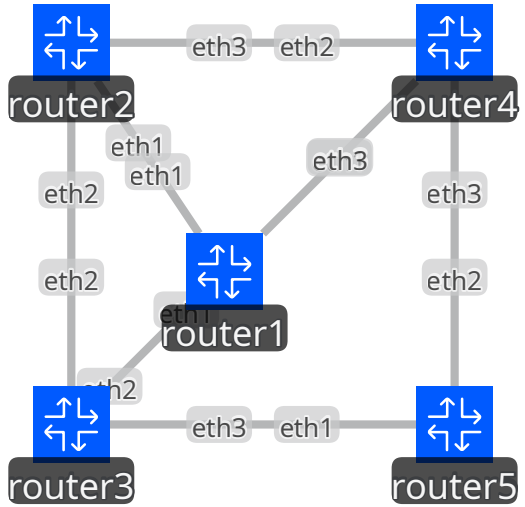
IS-IS (Intermediate System to Intermediate System)

- ISO/IEC 10589:2002
- Link-State routing protocol
- Widely used by ISP

PROSE evaluation

Collecting traces

- Emulation with containerlab
- Dataset with 120 packets
- All optional protocol features
- Same topology for IS-IS, EIGRP, Babel



Results

- Checksum, TLV, length field in header correctly found in the protocols
- Inference with TLV only works but for sub-TLV

Protocols	ISIS	Babel	EIGRP
Precision	68.57%	100%	100%
Recall	75%	88.89%	100%

MUSE evaluation

Setup

- Compare MUSE VS Random Fuzzer
- Line coverage measured on TLV parsing code

Results

- MUSE reaches higher median line coverage
- Random mutations rejected earlier

Packets	IS-IS		EIGRP		Babel	
	MUSE	Random Fuzzer	**MUSE**	Random Fuzzer	**MUSE**	Random Fuzzer
100	32.94	30.50	59.91	55.06	65.16	61.45
500	33.01	30.50	59.91	56.27	74.93	65.34
1000	35.04	32.73	59.91	56.68	76.70	71.97
2000	/	32.73	59.91	57.89	81.34	74.26

Results fuzzing on routing protocols

Test on FRRouting version 10.3

EIGRP

- Infinite loop
  - CPU saturated (~100%)
  - Hello packet with a valid Hello packet with valid TL + excessive padding (`0x00`) → loop increment = 0
- Read from unallocated memory
  - Potential crash, issue detected with Valgrind
  - Same problem but with other value of increment it access to unallocated memory
- Abort via assertion
  - Process restart
  - Two different assertions observed, likely same root cause

IS-IS

- Process restart with a specific input and crash after 2-3 seconds

=> Issues reported to FRRouting dev, two first already fixed

Limitations and potential extensions

- Not stream-based protocols
- More difficulties to infer sub-TLV
- Test on more protocols

Conclusion

Source code : https://github.com/Martinviv/fuzzing_routing_protocols

Tested routing protocols

- Developed **PROSE** tool
- Developed **MUSE** tool
- Inferred header, checksum and TLV structure
- Works successfully on EIGRP, IS-IS, Babel
- Found issues in open sources implementation of routing protocols

Future work

- Evaluate the tool on more protocols
- Extended for protocols bytestream like BGP

Routing

- Control plane
  - Runs routing protocols (OSPF,EIGRP, IS-IS…)
  - Calculates and updates the routing table
  - Responds to network changes
  - Determines the best path for traffic

- Data plane
  - Uses the routing table to forward packets