Componolit
Secure Systems Engineering

RecordFlux: Formal Message Specification and Generation of Verifiable Binary Parsers

Journée thématique du GT SSLR 2019

Alexander Senier

Paris, November 27th, 2019

RecordFlux
Communication Protocols

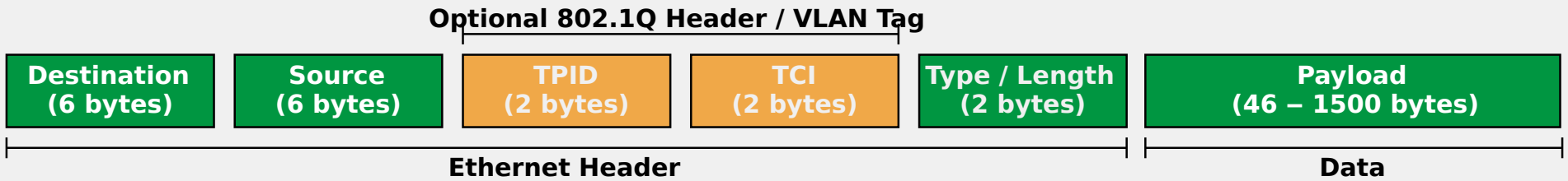
Componolit
Secure Systems Engineering

- **Vulnerabilites**

- **Critical Applications**
  - Automotive
  - Aviation
  - Critical infrastructure
  - Medicine

- Complex message formats
- No formal specifications
- Error-prone manual implementation
- Use of unsafe programming languages

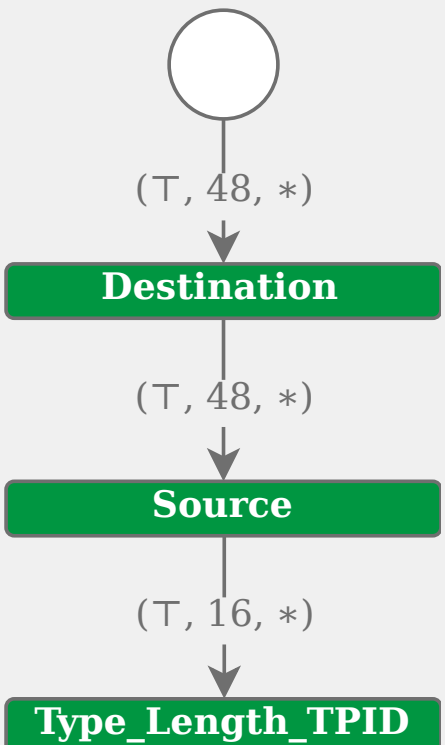
RecordFlux
Example: Ethernet Frame

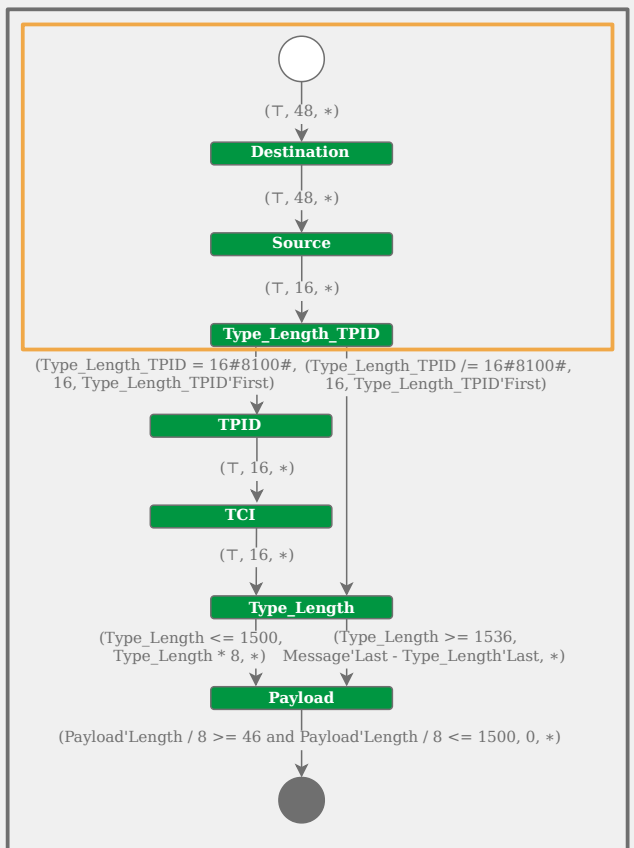


- Meaning of Type/Length field depends on its value
  - ≤ 1500: Length of Payload (IEEE 802.3)
  - ≥ 1536: Type of Payload (Ethernet II)
- Payload length depends on value of Type/Length field or on length of message
- Optional VLAN tag (IEEE 802.1Q)
  - Existence depends on the first two bytes of its own field
- VLAN tag and Type/Length field overlap

RecordFlux
Representation: Directed Acyclic Graph

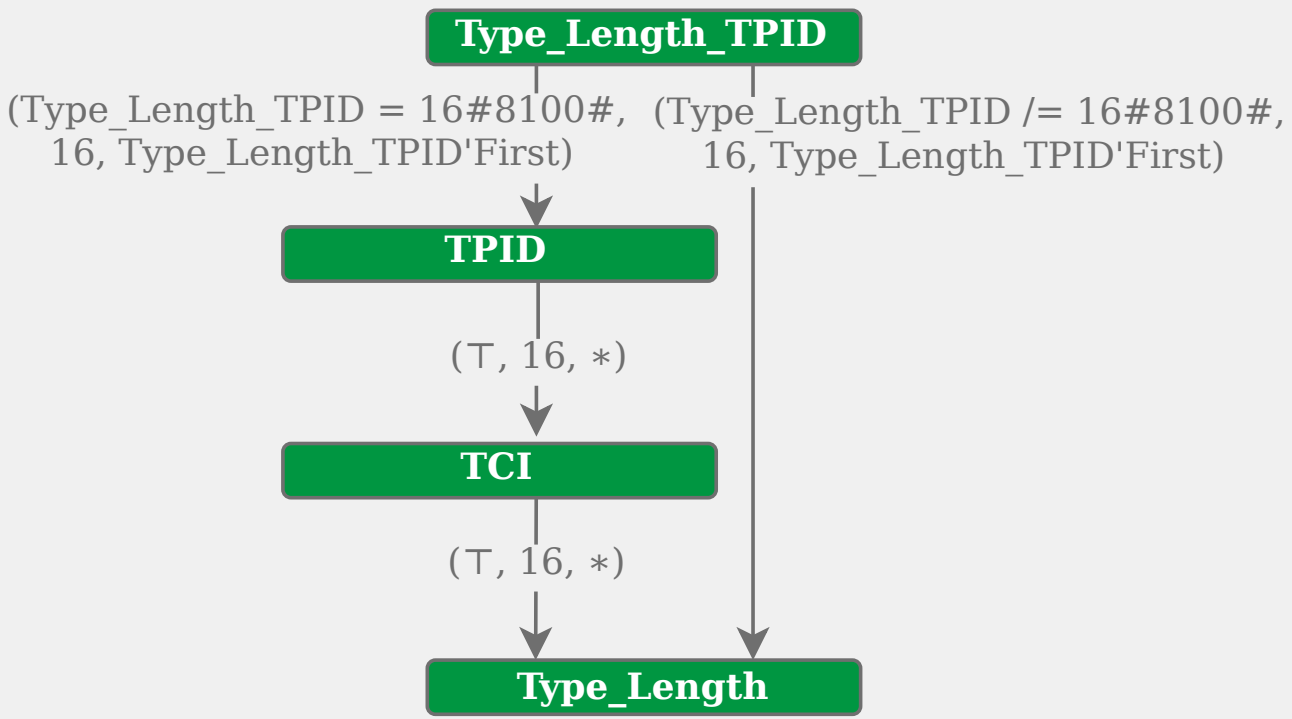
Componolit
Secure Systems Engineering

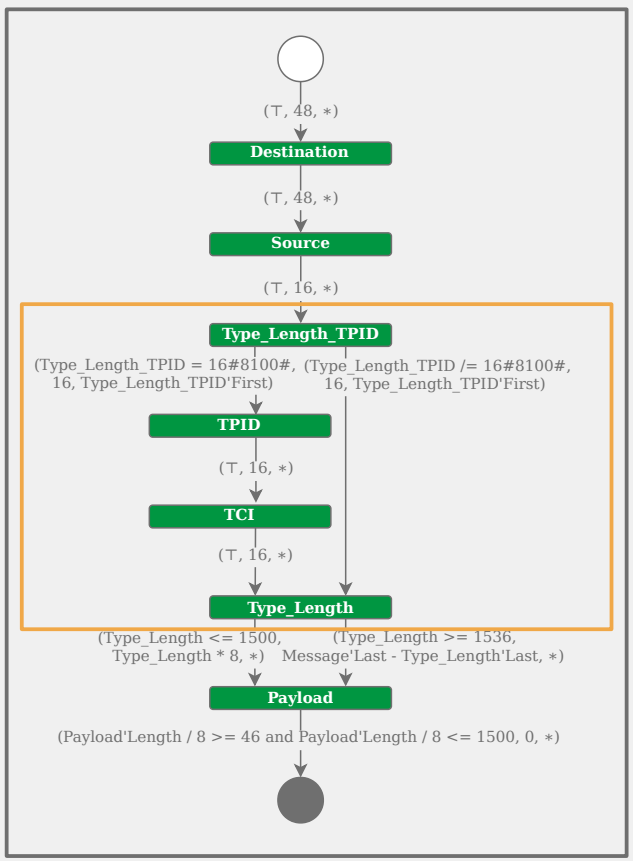




RecordFlux
Representation: Directed Acyclic Graph

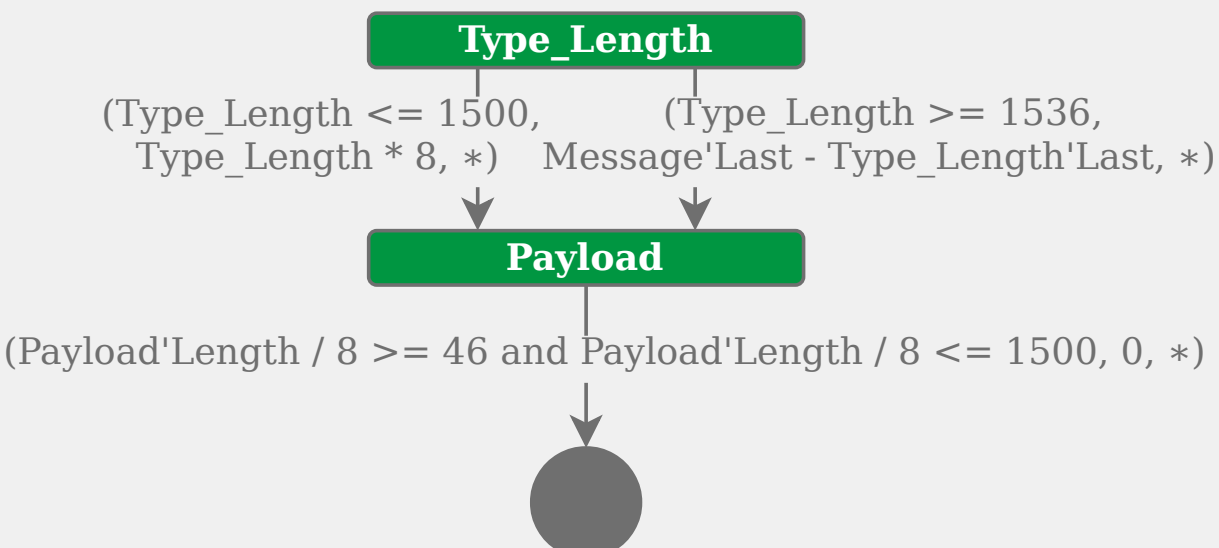
Componolit
Secure Systems Engineering

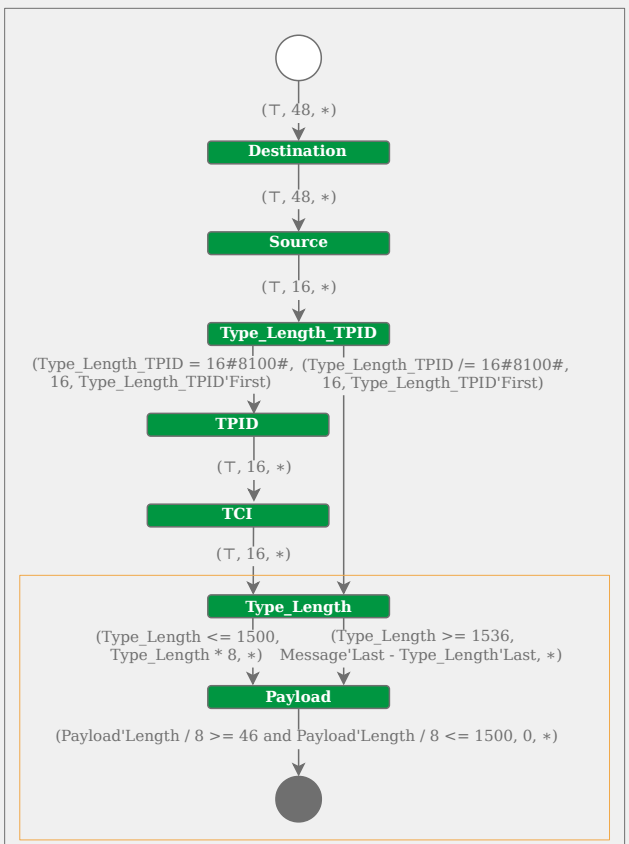




RecordFlux
Representation: Directed Acyclic Graph

Componolit
Secure Systems Engineering





RecordFlux
Specification Language

Componolit
Secure Systems Engineering

- **Order of Fields**
  - then
- **Length and Location**
  - with Length / with First
- **Field Condition**
  - if

```
package Ethernet is
   type Address is mod 2**48;
   type Type_Length is range 46 .. 2**16 - 1 with Size => 16;
   type Frame is message
         Destination : Address;
         Source      : Address;
         Type_Length : Type_Length
            then Payload
               with Length => Type_Length * 8
               if Type_Length <= 1500,
            then Payload
               with Length => Message'Last - Type_Length'Last
               if Type_Length >= 1536;
         Payload : Payload
            then null
               if Payload'Length / 8 >= 46
                  and Payload'Length / 8 <= 1500;
      end message;
end Ethernet;
```

Model Verification

- **Invariants proven for the model**
  - Field conditions are mutually exclusive
  - Field conditions do not contradict each other
  - Each field is reachable on at least one path from the initial node
  - Message fields are always located after the first message bit
  - Field length is never negative
  - Message fields cover all bits of a message on all paths
  - Overlaid fields are congruent with exactly one other field

RecordFlux
Code Generation

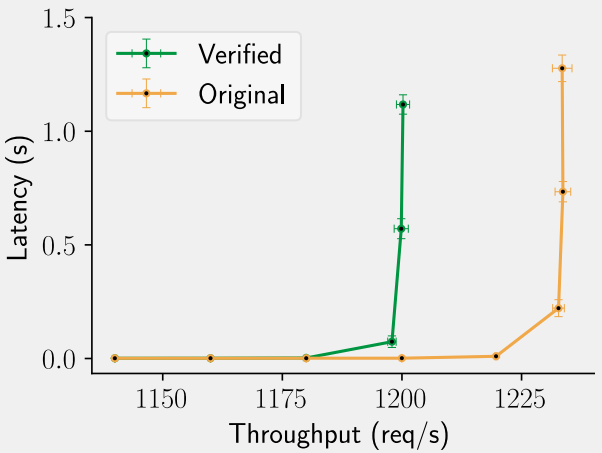
Componolit
Secure Systems Engineering

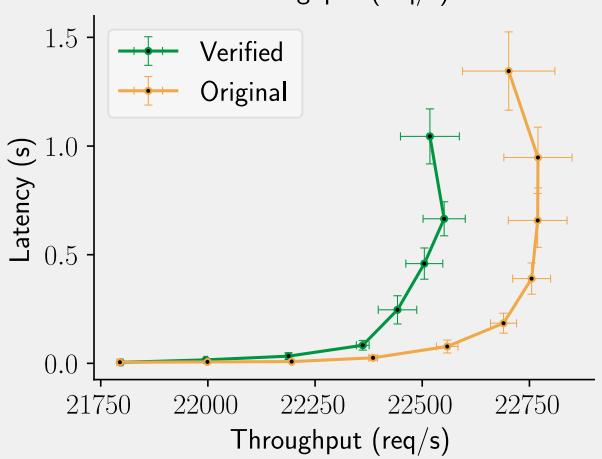
- **Code Generation**
  - SPARK code
  - Suitable for low-resource/embedded targets
  - Binary parser and generator
- **Properties**
  - Absence of runtime errors
  - Functional correctness

RecordFlux
Case Study: Verified TLS Parser

Componolit
Secure Systems Engineering

- Fizz – C++ TLS 1.3 implementation by Facebook[1]
- Fizz' parser replaced by SPARK implementation[2]
  - RecordFlux TLS 1.3 specification (RFC 8446)
  - Integration with C++ using manual glue code: mainly conversion between C++ data structures (e.g. vectors) and SPARK counterparts
- Absence of runtime errors proven for the parser
- Performance impact mostly due to conversion
  - Handshake: 2.7 % throughput loss (upper chart)
  - Record: 1.1 % throughput loss (lower chart)





[1]) https://github.com/facebookincubator/fizz [2]) https://github.com/Componolit/fizz

Summary

Componolit
Secure Systems Engineering

- **Source, Language Reference**
  - https://github.com/Componolit/RecordFlux
- **Work in progress**
  - Message sequences
  - Verified component-based TLS 1.3
- **Future work**
  - Protocol fuzzing
  - Cryptographic Proofs
  - Reverse Engineering
  - Traceability to informal specifications

Questions?

Componolit
Secure Systems Engineering

Alexander Senier
senier@componolit.com

@Componolit · componolit.com · github.com/Componolit