RecordFlux: Facilitating the Verification of Communication Protocols

Tobias Reiher
3 February 2019
Communication Protocols

Problems

- Implementing protocols is time-consuming and error prone
- Specifications available only in English prose
- Formal protocol specification rarely exist (or are feasible to do)
Communication Protocol Verification

Solution Space

- Security properties
  (security protocol proofs)

- Protocol semantics
  (temporal logic, model checking)

- Message formats
  (functional correctness)

- Absence of runtime errors
  (program verification)
High-assurance Implementation

SPARK

- **Programming language**
  - Imperative, object-oriented
  - Designed for error avoidance
  - Strong type system
  - Formal contracts

- **Verification toolset**
  - Data and control flow analysis
  - Dependency contracts
  - Absence of runtime errors
  - Functional correctness

- **Used in various critical and system-level projects**
  - Satellite software
  - Air Traffic Control
  - Secure workstation
  - Muen Separation Kernel (https://muen.sk)

- **More details on SPARK**
  - https://adacore.com/about-spark
High-assurance Implementation
A simple task: Calculating abs()

// Calculate absolute value of x
int abs_value(int x)
{
    if (x > 0) {
        return x;
    } else {
        return -x;
    }
}

// Let's try abs_value()
abs_value(-12345) ⟹ 12345
abs_value(56789) ⟹ 56789
abs_value(0) ⟹ 0
abs_value(-2147483648) ⟹ -2147483648
function Abs_Value (X : Integer) return Integer is
begin
  if X > 0 then
    return X;
  else
    return -X;
  end if;
end Abs_Value;

Proving...
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...
abs_value.adb:6:6: medium: overflow check might fail (e.g. when X = Integer'First) [possible explanation: subprogram at line 1 should mention X in a precondition]
High-assurance Implementation
Calculating abs() with SPARK

1 function Abs_Value (X : Integer) return Integer
2   with
3   Pre => X /= Integer'First;

Proving...
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...
Objective

- Dissection, generation and verification of communication protocols

Architecture
RecordFlux
Specification Language

- Type for messages
  - Message (message)
- Types for fields
  - Enumeration
  - Integer (range)
  - Composite (Payload_Type)
- Modularization
  - Package (package)

```plaintext
package TLS_Heartbeat is
  type Message_Type is (HEARTBEAT_REQUEST => 1,
                        HEARTBEAT_RESPONSE => 2)
    with Size => 8;
  type Length_Type is range 0 .. 2**14 - 20
    with Size => 16;
  type Heartbeat_Message is
    message
      Message_Type    : Message_Type;
      Payload_Length : Length_Type;
      Payload        : Payload_Type;
      Padding        : Payload_Type;
    end message;
end TLS_Heartbeat;
```

- Message Type (1 byte)
- Payload Length (2 bytes)
- Payload (0 .. 2**14-20 bytes)
- Padding (16 .. 2**14-20 bytes)

TLS Heartbeat Message (19 .. 2**14 bytes)
package TLS_Heartbeat is

  type Message_Type is (HEARTBEAT_REQUEST => 1, HEARTBEAT_RESPONSE => 2)
  with Size => 8;

  type Length_Type is range 0 .. 2**14 - 20 with Size => 16;

  type Heartbeat_Message is
    message
      Message_Type : Message_Type;
      Payload_Length : Length_Type
      then Payload
      with Length = Payload_Length * 8;
      Payload : Payload_Type
      then Padding
      with Length = Message'Last - Payload'Last;
      Padding : Payload_Type
      then null
      if Message'Length <= 2**14 * 8 and Padding'Length >= 16 * 8;
    end message;

end TLS_Heartbeat;
function Is_Contained (Buffer : Bytes) return Boolean
   with Ghost, Import;

procedure Label (Buffer : Bytes)
   with Post => Is_Contained (Buffer);

function Valid_Message_Type (Buffer : Bytes) return Boolean
   with Pre => Is_Contained (Buffer);

function Get_Message_Type (Buffer : Bytes) return Message_Type
   with Pre => (Is_Contained (Buffer) and then Valid_Message_Type (Buffer));

function Valid_Payload (Buffer : Bytes) return Boolean
   with Pre => Is_Contained (Buffer);

procedure Get_Payload (Buffer : Bytes; First : out Natural; Last : out Natural)
   with Pre => (Is_Contained (Buffer) and then Valid_Payload (Buffer)),
   Post => (First = Get_Payload_First (Buffer) and then
    Last = Get_Payload_Last (Buffer));

function Is_Valid (Buffer : Bytes) return Boolean
   with Pre => Is_Contained (Buffer);
RecordFlux
Using the Generated Code

with TLS.Heartbeat_Message; use TLS.Heartbeat_Message;

procedure Main is
  Buffer : Bytes := Read;
  Tag    : Message_Type;
  First  : Natural;
  Last   : Natural;
begin
  Label (Buffer);
  Tag := Get_Message_Type (Buffer);
  Get_Payload (Buffer, First, Last);
end Main;

Proving...
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...
tls-test.adb:10:6: medium: precondition might fail
tls-test.adb:12:6: medium: precondition might fail
**RecordFlux**

**Using the Generated Code**

```plaintext
with TLS.Heartbeat_Message; use TLS.Heartbeat_Message;

procedure Main is
  Buffer : Bytes := Read;
  Tag    : Message_Type;
  First  : Natural;
  Last   : Natural;
begin
  Label (Buffer);
  if Is_Valid (Buffer) then
    Tag := Get_Message_Type (Buffer);
    Get_Payload (Buffer, First, Last);
  end if;
end Main;

Proving...
Phase 1 of 2: generation of Global contracts ...
Phase 2 of 2: flow analysis and proof ...
```
RecordFlux
Current State

- **Security properties**
  (security protocol proofs)
  - *Future work*

- **Protocol semantics**
  (temporal logic, model checking)
  - *Future RecordFlux version*

- **Message formats**
  (functional correctness)
  - *Addressed by RecordFlux*

- **Absence of runtime errors**
  (program verification)
  - *Addressed by SPARK*
RecordFlux
Application: GreenTLS

- Component-based high-assurance implementation of TLS 1.3
- Started this year, partially funded by European Union and state of Saxony
- Critical components in SPARK using RecordFlux
- Genode OS Framework as a base platform, but potentially others
- Challenges
  - How to separate architecture (without breaking it)?
  - How can security be proven?
  - Performance vs. side channel avoidance?
  - ...
- Source available on GitHub: https://github.com/Componolit/GreenTLS
RecordFlux

Conclusion

■ Current state

 ▪ Specification language powerful enough to specify real-world binary protocols: Ethernet, IPv4, UDP, and TLS 1.3 (in progress)
 ▪ Generation of SPARK 2014 code: Absence of runtime errors and correctness of message formats

■ Next steps

 ▪ Message generation
 ▪ More protocols (e.g., USB)
 ▪ Non-TLV message schemes

■ Source code

 ▪ Available on GitHub: https://github.com/Componolit/RecordFlux
Questions?

Tobias Reiher
reiher@componolit.com

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