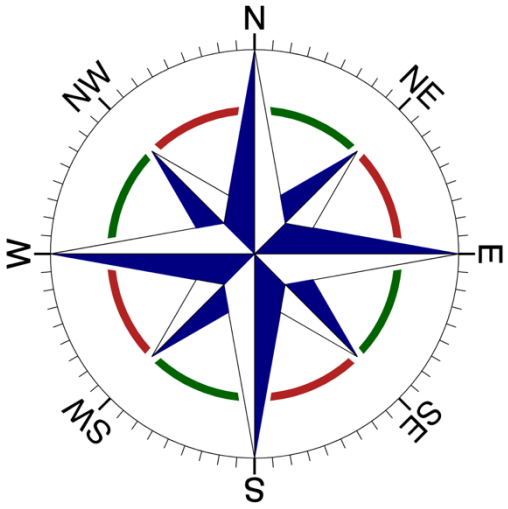
Specifying Constraints in SCL5 for Intrusion Detection

A Technical Report

By Fahim Imam



COPASSTR20-1
Compass Group, Queen's University
Kingston, ON, Canada
February 2020

Table of Contents

SPECIFYING CONSTRAINTS IN SCL5 FOR INTRUSION DETECTION 3

Background 3
Constraint Specification 3
Single-Packet Constraints 4
Multi-Packet Constraints 5
Handling Multicast and Unicast Addressing 6

EXAMPLE CONSTRAINTS 8

NFS and NFSACL 8
RTPS.AckNack 10
RTPS.Data(P) 10
RTPS.Data 11
RTPS.Data(R) 12
RTPS.Data(W) 13
RTPS.GAP 14
RTPS.HEARTBEAT 14
RTPS.INFO_DST 15
RTPS.PING 15

Specifying Constraints in SCL5 for Intrusion Detection

Last edited by **Fahim Imam (fahim.imam@queensu.ca)**

Background

The goal of our IDS is to check the current packet from a stream of packets for a set of conditions. We refer to such conditions as constraints. The idea is to validate if the current packet along with the information it contains is allowed to be there in the stream. The constraint engine checks each of the packets in a sequence and inspects the latter based on a set of known constraints. We describe a proposed approach of specifying constraints in SCL5.

Constraint Specification

At the simplest level, we are essentially talking about specifying the behavior of an IDS where the input of the system is the next incoming packet which has to go through a set of inspections to prove its validity. These inspections are needed to be specified as a set of constraints written by a Network Engineer who has the familiarity with the normal behavior of the network along with its associated protocols.
IDS Input: The Next Incoming Packet. **IDS Output:** Validity Status of the Input Packet.

Ideally, each of the incoming Packet types must have a set of constraints for which there is a known set of information to compare. This information may include a particular content of the incoming packet or a sequence of packets that must exist in a particular order for the incoming packet to be valid.

Following convention is proposed so that we can specify the type of packets and their contents in a uniform way independent of the protocol specification. For example, the IGMP and RTPS protocols have different specification and naming conventions for their packet types or submessage types and the field values. We follow the following convention:

`PROTOCOL-TYPE.PACKET-TYPE.FIELD-TYPE`.

However, the protocol type can be safely ignored for the protocol-specific packet types that are locally declared within the `SCL5` file. For example, when writing a constraint for the `DATA` packet within the `RTPS.SCL5` file, we do not need to specify the RTPS Protocol-Type to access the fields of the target packet (e.g., we can simply write `DATA.SrcIP` or `DATA.DstIP`). Accessing the appropriate field value for a packet or a submessage type using the dot operators must conform to the SCL5 specification of the protocols.

The protocol type is required explicitly for the packets that are specified in an external SCL5 document. For example, if the constraint for the DATA packet within the `RTPS.SCL5` requires a packet type from the `IGMP.SCL5` then we have to explicitly specify the protocol type for that IGMP packet (e.g., `IGMP.MembershipReport.ScrIP`). The constraints should be written immediately next to the target packet type specification within the constraints block of the protocol specific SCL5 specification.

```
<constraints>
        <constraint>
            CSL Code
        </constraint>
        <constraint>
            CSL Code
        </constraint>
...
</constraints>
```

There are two basic kinds of constraints: `MULTI-PACKET`and `SINGLE-PACKET`.

Single-Packet Constraints

The single packet constraints are used to verify the known static environmental information or facts within a single incoming packet. We simply need to specify the constraint type using `TYPE: SINGLE-PACKET-ENV` followed by the `VALID-ENV` that includes the list of field types of the target packet (prefixed with `@`) that are expected to have known environmental values. Following is the generic template for a single packet environmental constraint.

```
1. <constraint>
2.   TYPE: SINGLE-PACKET-ENV
3.   VALID-ENV: @PACEKT-X (FIELD-1, FIELD-2, ..., FIELD-N)
4. </constraint>
```

Example. Following is an example of a single-packet constraint where we are interested in validating the target packet, the RTPS participant packet (`RTPS.DATA_P`), to be from a known source IP (`SrcIP`), destination IP (`DstIP`), and destination port (`DstPort`).

```
1. <constraint>
2.      TYPE: SINGLE-PACKET-ENV
3.      VALID-ENV: @RTPS.DATA_P (SrcIP, DstIP, DstPort)
4. </constraint>
```

Multi-Packet Constraints

A multipacket constraint is a constraint which requires information from more than one packet in a specified sequence of packet types. This kind of constraints requires the following set of specifications against a target packet type within the `SCL5` document.

- Specify the Constraint Type, `TYPE` relevant to the constraint
- Specify the set of Valid Sequences, `VALID-SEQ` of the Related Packets
- Specify the Evaluation Statements for the constraint within the curly braces.

A constraint for the target packet type Packet-X can be written using the following template where packet types prefixed with `(1)..(3)` are the sequence of packets until the target packet type prefixed with `@`. The final packet type prefixed with `~` is an optional packet which is only required for some of the constraints which will be explained through examples later.

```
<constraint>
1. TYPE: MULTI-PACKET
2. VALID-SEQ: (1)PACKET-A, (2)PACKET-B, (3)PACKET-B, .., @PACKET-X, ~PACKET-Z
3. {
4.   (1)PACKET-A.FIELD1 == (2)PACKET-B.FIELD1
5.   @PACKET-X.FIELD1 == (3)PACKET-B.FIELD1
6.   @PACKET-X.FIELD2 == (2)PACKET-B.FIELD2
7. }
</constraint>
```

It should be noted that a target packet can have a same type of packets in the predefined sequence appearing multiple times. For example, the packets `(2)PACKET-B` and `(3)PACKET-B` above are both referring to the same type of packet with a different sequence. Also, it should be noted that a multipacket constraint can have more than one valid sequences of packets as will be exemplified later.

Example. Following constraint specifies that a publisher (`RTPS.DATA_W`) must be a valid participant (`RTPS.DATA_P`) in RTPS protocol. Notice that in order for this latter constraint to satisfy, a participant must correspond to a valid IGMP membership report (`IGMP.V2Report` packet) as specified in line 8.

```
1. <constraint>
2. TYPE: MULTI-PACKET
3. VALID-SEQ: (1)IGMP.V2Report, (2)RTPS.DATA_P, @RTPS.DATA_W, ~IGMP.V2LEAVE
4.  {
5.      @RTPS.DATA_W.SrcIP == (2)RTPS.DATA_P.SrcIP
6.      @RTPS.DATA_W.DstIP == (2)RTPS.DATA_P.DstIP
7.      @RTPS.DATA_W.DstPort == (2)RTPS.DATA_P.DstPort
8.      @RTPS.DATA_W.DstIP == (1)IGMP.V2Report.groupAddr
9.  }
10.</constraint>
```

Handling Multicast and Unicast Addressing

The multi-packet constraints can be classified into three specific types: (a) Multi-Packet Multicast, (b) Multi-Packet Unicast, (c) Multi-Packet Broadcast. A multipacket constraint that requires checking the unicast or multicast addressing for its particular packet types can be specified using the predefined boolean functions called the `unicast` or the `multicast` within the `IF` statement immediately after the `VALID-SEQ` statement. The input parameter for the `unicast` or the `multicast` functions must be the destination IP address (`DstIP`) of the required packets.

```
1. <constraint>
2. TYPE: MULTI-PACKET
3. VALID-SEQ: (1)RTPS.DATA_R, (2)RTPS.DATA_W, @RTPS.DATA, ~IGMP.V2LEAVE
4. IF:  unicast(@RTPS.DATA.DstIP) AND unicast((1)RTPS.DATA_R.DstIP) AND
5.      unicast((2)RTPS.DATA_W.DstIP)
6.       {
7.        (1)RTPS.DATA_R.DstIP == (2)RTPS.DATA_W.SrcIP
8.        (1)RTPS.DATA_R.SrcIP == (2)RTPS.DATA_W.DstIP
9.        (2)RTPS.DATA_W.TopicName == @RTPS.DATA_R.TopicName
```

```
10.       @RTPS.DATA.WriterEntityID ==
11.         (2)RTPS.DATA_W.serializedData.topicdata.ENDPOINTGUID.EntityID
12.      }
13. </constraint>
```

```
1. <constraint>
2. TYPE: MULTI-PACKET
3. VALID-SEQ: (1)RTPS.DATA_W, @RTPS.DATA
4. IF: multicast(@RTPS.DATA.DstIP) AND multicast((1)RTPS.DATA_W.DstIP)
5. {
6.      @RTPS.DATA.SrcIP == (1)RTPS.DATA_W.SrcIP
7.      @RTPS.DATA.DstIP == (1)RTPS.DATA_W.DstIP
8.      @RTPS.DATA.writerEntity.key ==
9.       (1)RTPS.DATA_W.serializedData.topicData.PIDENDPOINTGUID.entityid.key
10. @RTPS.DATA.writerEntity.kind ==
11.     (1)RTPS.DATA_R.serializedData.topicData.PIDENDPOINTGUID.entityid.kind
12. }
13. </constraint>
```

EXAMPLE CONSTRAINTS

NFS and NFSACL

```
<constraints>
  <constraint>
    TYPE: MULTI-PACKET
    VALID-SEQ: (1)NFSACL, @NFS
    IF: @NFS.MessageType == "Call"
    {
      @NFS.SrcIP == (1)NFSACL.SrcIP
      @NFS.DstIP == (1)NFSACL.DstIP
    }
  </constraint>
  <constraint>
   TYPE: MULTI-PACKET
   VALID-SEQ: (1)NFS, (2)NFSACL, @NFS
   IF: (1)NFS.MessageType == "Call" AND @NFS.MessageType == "Reply"
   {
     @NFS.SrcIP == (2)NFSACL.SrcIP
     @NFS.DstIP == (2)NFSACL.DstIP
   }
  </constraint>
</constraints>
```

```
<constraints>
  <constraint>
    TYPE: SINGLE-PACKET-ENV
    VALID-ENV: @NFSACL(SrcIP, DstIP)
  </constraint>

  <constraint>
    TYPE: MULTI-PACKET
    VALID-SEQ: (1)NFSACL, @NFSACL
    IF: @NFSACL.MessageType == "Reply" AND (1)NFSACL.MessageType == "Call"
    {
       @NFSACL.SrcIP == (1)NFSACL.DstIP
       @NFSACL.DstIP == (1)NFSACL.SrcIP
    }
  </constraint>
</constraints>
```

RTPS Constraints

Constraints for different RTPS sub-message types.

RTPS.AckNack

```
<constraints>
        <constraint>
        TYPE: MULTI-PACKET
        VALID-SEQ: (1)RTPS.DATA_P, @RTPS.ACKNACK
        {
         @RTPS.ACKNACK.SrcIP == (1)RTPS.DATA_P.SrcIP
         @RTPS.ACKNACK.DstIP == (1)RTPS.DATA_P.DstIP
         @RTPS.ACKNACK.writerEntityKind == (1)RTPS.DATA_P.writerEntityKind
        }
        </constraint>
</constraints>
```

RTPS.Data(P)

```
<constraints>
 <constraint>
        TYPE: SINGLE-PACKET-ENV
        VALID-ENV: @RTPS.DATA_P (SrcIP, DstIP, DstPort)
 </constraint>
</constraints>
```

RTPS.Data

```
<constraints>
        <constraint>
                TYPE: MULTI-PACKET --multicust
                VALID-SEQ: (1)RTPS.DATA_W, @RTPS.DATA

IF: multicast(@RTPS.DATA.DstIP) AND multicast((1)RTPS.DATA_W.DstIP)
{
                    @RTPS.DATA.SrcIP == (1)RTPS.DATA_W.SrcIP
                    @RTPS.DATA.DstIP == (1)RTPS.DATA_W.DstIP
                    @RTPS.DATA.writerEntity.key ==
(1)RTPS.DATA_W.serializedData.topicData.PIDENDPOINTGUID.entityid.key
                    @RTPS.DATA.writerEntity.kind ==
(1)RTPS.DATA_R.serializedData.topicData.PIDENDPOINTGUID.entityid.kind
                }
        </constraint>

        <constraint> -- May not be necessary
                TYPE: MULTI-PACKET
                VALID-SEQ: (1)RTPS.DATA_R, @RTPS.DATA
                {
                    @RTPS.DATA.SrcIP == (1)RTPS.DATA_R.SrcIP
                @RTPS.DATA.DstIP == (1)RTPS.DATA_R.DstIP
                -- Normalization: Transform from @RTPS.DATA.writerEntity ==
(1)RTPS.DATA_R.serializedData.topicData.PIDENDPOINTGUID.entityid
                @RTPS.DATA.writerEntity.key ==
(1)RTPS.DATA_R.serializedData.topicData.PIDENDPOINTGUID.entityid.key
                @RTPS.DATA.writerEntity.kind ==
(1)RTPS.DATA_R.serializedData.topicData.PIDENDPOINTGUID.entityid.kind
                 }
        </constraint>

        <constraint>
                TYPE: MULTI-PACKET --unicast
                -- CONFLICT:  1
                VALID-SEQ: (1)RTPS.DATA_R, (2)RTPS.DATA_W, @RTPS.DATA,
~IGMP.V2LEAVE
                IF:  unicast(@RTPS.DATA.DstIP) AND
                     unicast((1)RTPS.DATA_R.DstIP) AND
                     unicast((2)RTPS.DATA_W.DstIP)    --IF: is an optional
statement after the VALID-SEQ
                {
                (1)DATA_R.DstIP == (2)DATA_W.SrcIP
                (1)DATA_R.SrcIP == (2)DATA_W.DstIP
                (2)DATA_W.TopicName == @DATA_R.TopicName
                @DATA.WriterEntityID ==
(2)DATA_W.serializedData.topicdata.ENDPOINTGUID.EntityID
            }
        -- </constraint>
```

```
        -- <constraint>
        --      TYPE: MULTI-PACKET --unicast
        --      CONFLICT:  1
                VALID-SEQ: (1)RTPS.DATA_W, (2)RTPS.DATA_R, @RTPS.DATA,
~IGMP.V2LEAVE
                {
                    (1)DATA_W.SrcIP == (2)DATA_R.DstIP
                    (2)DATA_R.SrcIP == (1)DATA_W.DstIP
                    (1)DATA_W.TopicName == (2)DATA_R.TopicName
                    @DATA.WriterEntityID ==
(2)DATA_W.serializedData.topicdata.ENDPOINTGUID.EntityID
                }
        </constraint>
</constraints>
```

RTPS.Data(R)

```
<constraints>
        <constraint>
                TYPE: MULTI-PACKET
                VALID-SEQ: (1)IGMP.V2Report, (2)RTPS.DATA_P, @RTPS.DATA_R,
~IGMP.V2Leave -- ~IGMP.V3leave
                {
                (1)IGMP.REPORT.groupAddressIP == (2)RTPS.DATA_P.DstIP
                @RTPS.DATA_R.SrcIP == (2)RTPS.DATA_P.SrcIP
                @RTPS.DATA_R.DstIP == (2)RTPS.DATA_P.DstIP

                ~IGMP.V2Leave.SrcIP == (1)IGMP.V2Report.SrcIP
                ~IGMP.V2Leave.groupAddress == (1)IGMP.V2Reporet.GroupAddressIp

                IGMP.V3Report.groupRecordInfo.recordType == 1
                IGMP.V3Report.SrcIP==IGMP.V2Report.SrcIP
                IGMP.V3Report.groupRecrodInfo==IGMP.V2Report.GroupAddressIP
                }
        </constraint>
```

```
<constraint>
                TYPE: MULTI-PACKET
VALID-SEQ: (1)IGMP.V3Report, (2)RTPS.DATA_P, @RTPS.DATA_R,    ~IGMP.V2Leave -
- ~IGMP.V3leave
                {
                        (1)IGMP.V3Report.groupRecordInfo.recordType == 2
                        (1)IGMP.V3Report.groupAddressIP == RTPS.DATA_P.DstIP
                        (1)IGMP.V3ReportSrcIP == RTPS.DATA_P.SrcIP
                        @RTPS.DATA_R.SrcIP == RTPS.DATA_P.SrcIP
                        @RTPS.DATA_R.DstIP == RTPS.DATA_P.DstIP
                }
        </constraint>
</constraints>
```

RTPS.Data(W)

```
<constraints>
        <constraint>
                TYPE:SINGLE-PACKET-ENV
                VALID-ENV: @RTPS.DATA_W (SrcIP,
serializedData.topicData.PIDTOPICNAME.topicName.name,
serializedData.topicData.pidTypeName.typeName,
serializedData.topicData.pidReliability.reliabilityQos)
        </constraint>
        <constraint>
                TYPE: MULTI-PACKET
                VALID-SEQ: (1)IGMP.V2Report, (2)RTPS.DATA_P, @RTPS.DATA_W,
~IGMP.V2LEAVE
                {
                        @RTPS.DATA_W.SrcIP == (2)RTPS.DATA_P.SrcIP
                        @RTPS.DATA_W.DstIP == (2)RTPS.DATA_P.DstIP
                        @RTPS.DATA_W.DstPort == (2)RTPS.DATA_P.DstPort
```

```
                @RTPS.DATA_W.DstIP == (1)IGMP.V2Report.groupAddr
        }
    </constraint>
</constraints>
```

RTPS.GAP

```
<constraints>
    <constraint>
    TYPE: MULTI-PACKET
    VALID-SEQ: (1)RTPS.DATA_W, @RTPS.GAP
    {
        @RTPS.GAP.SrcIP == (1)RTPS.DATA_W.SrcIP
        @RTPS.GAP.writerEntityKind == (1)RTPS.DATA_W.writerEntityKind
    }
    </constraint>
</constraints>
```

RTPS.HEARTBEAT

```
<constraints>
    <constraint>
    TYPE: MULTI-PACKET
    VALID-SEQ: (1)RTPS.DATA_P, @RTPS.HEARTBEAT
    {
        @RTPS.HEARTBEAT.SrcIP == (1)RTPS.DATA_P.SrcIP
        @RTPS.HEARTBEAT.DstIP == (1)RTPS.DATA_P.DstIP
        @RTPS.HEARTBEAT.writerEntityKind ==
(1)RTPS.DATA_P.writerEntityKind
    }
    </constraint>
</constraints>
```

RTPS.INFO_DST

```
<constraints>
        <constraint>
        TYPE: MULTI-PACKET
        VALID-SEQ: (1)RTPS.DATA_P, @RTPS.INFO_DST
        {
                @RTPS.INFO_DST.SrcIP == (1)RTPS.DATA_P.SrcIP
                @RTPS.INFO_DST.DstIP == (1)RTPS.DATA_P.DstIP
                @RTPS.INFO_DST.hostId == (1)RTPS.DATA_P.hostId
        }
        </constraint>
</constraints>
```

RTPS.PING

```
<constraints>
        <constraint>
                TYPE: SINGLE-PACKET-ENV
                VALID-ENV: @RTPS.PING (SrcIP, DstIP)
        </constraint>
        <constraint>
        TYPE: MULTI-PACKET
        VALID-SEQ: (1)RTPS.DATA_P, @RTPS.PING
        {
                @RTPS.PING.writerEntityKind == (1)RTPS.DATA_P.writerEntityKind
        }
        </constraint>
</constraints>
```