black hat EUROPE 2021

november 10-11, 2021

BRIEFINGS

The Data Distribution Service (DDS) Protocol is Critical Let's Use it Securely!

Ta-Lun Yen — **TXOne Networks**

Federico Maggi — **Trend Micro Research**

Erik Boasson — **ADLINK Tech. Labs**

C. Toyama, P. Kuo, and M. Cheng — TXOne Networks

Víctor Mayoral Vilches — Alias Robotics

txOne networks

TREND MICRO | research

ADLINK Leading EDGE COMPUTING

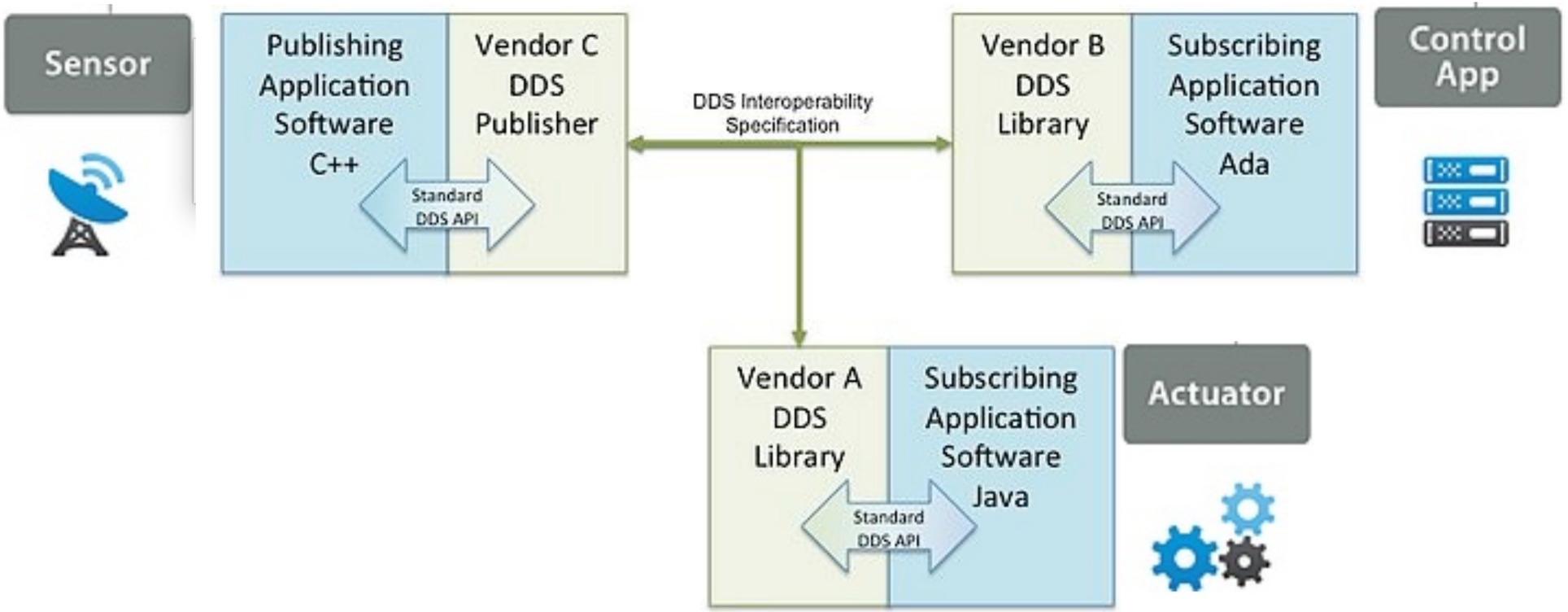
ALIAS ROBOTICS Robot Cybersecurity

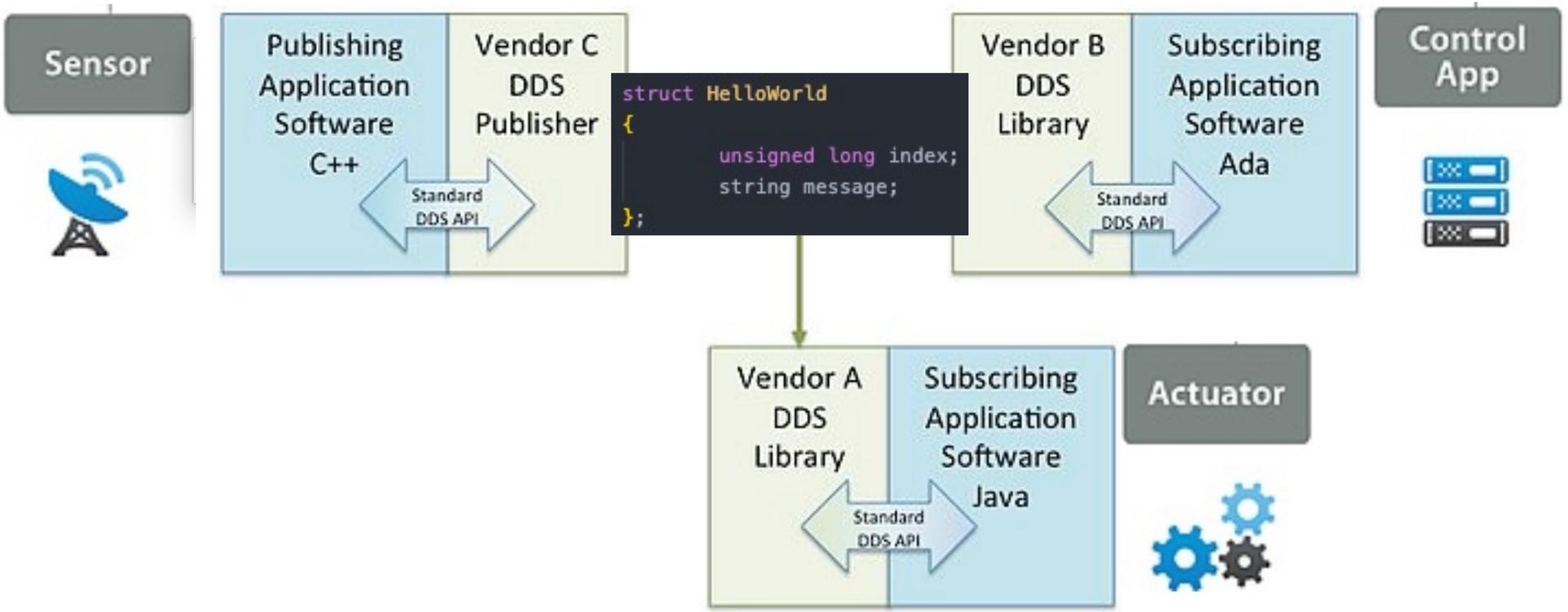
black hat EUROPE 2021
november 10-11, 2021
BRIEFINGS

What's DDS and why you may not have heard of it

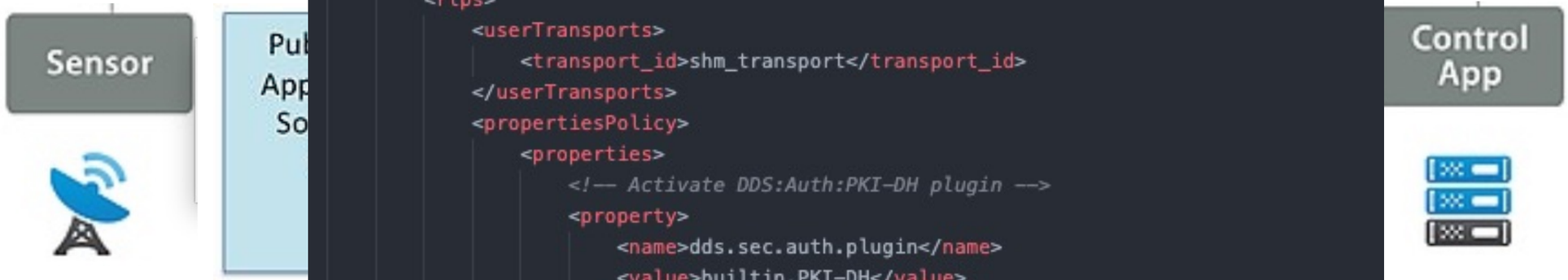
Erik Boasson ADLINK Tech. Labs


Sensor
Control App
Actuator


Sensor
Publishing Application Software C++
Vendor C DDS Publisher
Standard DDS API
DDS Interoperability Specification
Vendor B DDS Library
Subscribing Application Software Ada
Standard DDS API
Control App
Vendor A DDS Library
Subscribing Application Software Java
Standard DDS API
Actuator


Sensor
Publishing Application Software C++
Vendor C DDS Publisher
Standard DDS API
struct HelloWorld
{
unsigned long index;
string message;
};
Vendor B DDS Library
Subscribing Application Software Ada
Standard DDS API
Control App
Vendor A DDS Library
Subscribing Application Software Java
Standard DDS API
Actuator

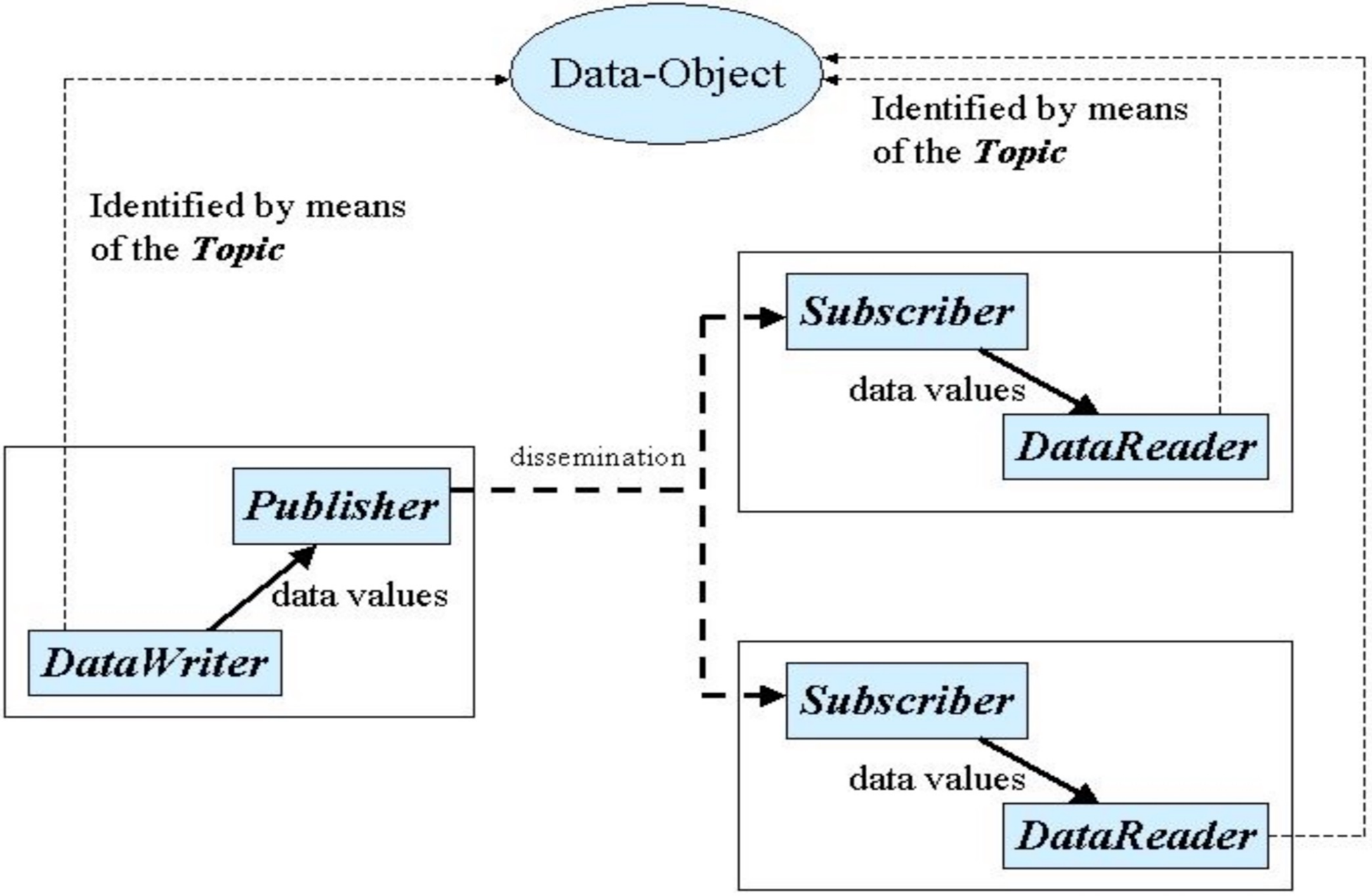

DDS
Sensor
enum MyEnum
{
A,
B,
C
};
enum MyBadEnum
{
A1,
B1,
C1
};
struct MyEnumStruct
{
MyEnum my_enum;
};
struct MyBadEnumStruct
{
MyBadEnum my_enum;
};
struct StringStruct
{
string my_string;
};
struct LargeStringStruct
{
string<41925> my_large_string;
};
// WString bounds
struct WStringStruct
{
wstring my_wstring;
};
struct LargeWStringStruct
{
wstring<41925> my_large_wstring;
};
struct NewAliases
{
int8 int8_;
uint8 uint8_;
int16 int16_;
uint16 uint16_;
int32 int32_;
uint32 uint32_;
int64 int64_;
@default(555) uint64 uint64_;
@non_serialized string local_string;
};
union WCharUnion switch (wchar)
{
case 'a':
long case_zero;
case 'b':
long case_one;
};




```
<profiles>
    <transport_descriptors>
        <transport_descriptor>
            <transport_id>shm_transport</transport_id>
            <type>SHM</type>
        </transport_descriptor>
    </transport_descriptors>

    <participant profile_name="secure_participant_profile" is_default_profile="true">
        <rtps>
            <userTransports>
                <transport_id>shm_transport</transport_id>
            </userTransports>
            <propertiesPolicy>
                <properties>
                    <!-- Activate DDS:Auth:PKI-DH plugin -->
                    <property>
                        <name>dds.sec.auth.plugin</name>
                        <value>builtin.PKI-DH</value>
                    </property>
                    <!-- Configure DDS:Auth:PKI-DH plugin -->
                    <property>
                        <name>dds.sec.auth.builtin.PKI-DH.identity_ca</name>
                        <value>file://maincacert.pem</value>
                    </property>
                    <property>
                        <name>dds.sec.auth.builtin.PKI-DH.identity_certificate</name>
                        <value>file://mainsubcert.pem</value>
                    </property>
                    <property>
                        <name>dds.sec.auth.builtin.PKI-DH.private_key</name>
                        <value>file://mainsubkey.pem</value>
                    </property>
```

```cpp
using namespace org::eclipse::cyclonedds;

int main()
{
  dds::domain::DomainParticipant participant(0);
  dds::pub::Publisher publisher(participant);
  dds::topic::Topic<HelloWorld> topic(participant, "HelloWorld");
  dds::pub::DataWriter<HelloWorld> writer(publisher, topic);

  unsigned i = 0;
  while (true)
  {
    HelloWorld msg(i++, "Hello world!");
    writer << msg;
    std::this_thread::sleep_for(std::chrono::seconds(1));
  }
  return 0;
}
```

```
struct HelloWorld
{
    unsigned long index;
    string message;
};
```


Data-Object
Identified by means of the Topic
Identified by means of the Topic
Subscriber
data values
DataReader
dissemination
Publisher
data values
DataWriter
Subscriber
data values
DataReader

From standard to implementations

DDS

Very open source

ADLINK
Leading EDGE COMPUTING

ECLIPSE FOUNDATION

eProsima
The Middleware Experts

OCI
WE ARE SOFTWARE ENGINEERS.

Mostly closed source

TWINOAKS COMPUTING INC.
PRACTICAL MIDDLEWARE EXPERTISE

Gurum Networks

SPARX SYSTEMS

Mostly closed source, with open-source integrations

rti
Your systems. Working as one.

Transportation

PRORAIL

ProRail

Large Scale Rail Network Management System

DDS provides ProRail with a reliable, real-time and fault-tolerant data-sharing platform to manage critical information within the Dutch railway network. With more than six thousand trains and 1.2 million passengers travelling on it on a daily basis, the Netherlands railway system is one of the busiest in Europe. Read more

NAV CANADA

NAV CANADA uses DDS to run real-time traffic management in the world's second busiest air space

Air Traffic Control in Canada has run on DDS since 2014. The Canadian Automated Air Traffic System (NAVCANtrac) system manages 3.3 million flights over 18 million square kilometers and is the second largest air navigation service providerÂ in the world. It uses DDS to manage, automate and integrate flight data between NAV CANADA air traffic control facilities, multiple surveillance sources, and external systems – safely, securely and with 24/7 reliability.

Coflight

Advanced Flight Data Processors Used in European Air Traffic Management System

Coflight is developing the most advanced Flight Data Processor (FDP) in the world and its DDS based architecture helps guarantee the long-term extensibility, scalability and availability of an Air Traffics Management (ATM) system that will be used to optimize air space usage, reduce the environmental impact of aviation and improve flight cost efficiency.

NAV CANADA improves air traffic management with RTI platform

8 September 2013 (Last Updated September 8th, 2013 18:30)

NAV CANADA is set to improve its NAVCANtrac air traffic management (ATM) system with Real-Time Innovations (RTI) Connext Data Distribution Service (DDS) middleware.

Share Article

NAV CANADA is set to improve its NAVCANtrac air traffic management (ATM) system with Real-Time Innovations (RTI) Connext Data Distribution Service (DDS) middleware.

NAV Canada said that it selected the RTI middleware platform after a successful implementation of the evaluation stage in order to replace the first-generation distribution architecture of its NAVCANtrac ATM system.

Top 5 M

Analysis

How 3D v could hel challenge

News 4 da

EASA: Bo ungroun

https://archive.is/wip/xAy2b

Autonomous Vehicles

VOLKSWAGEN

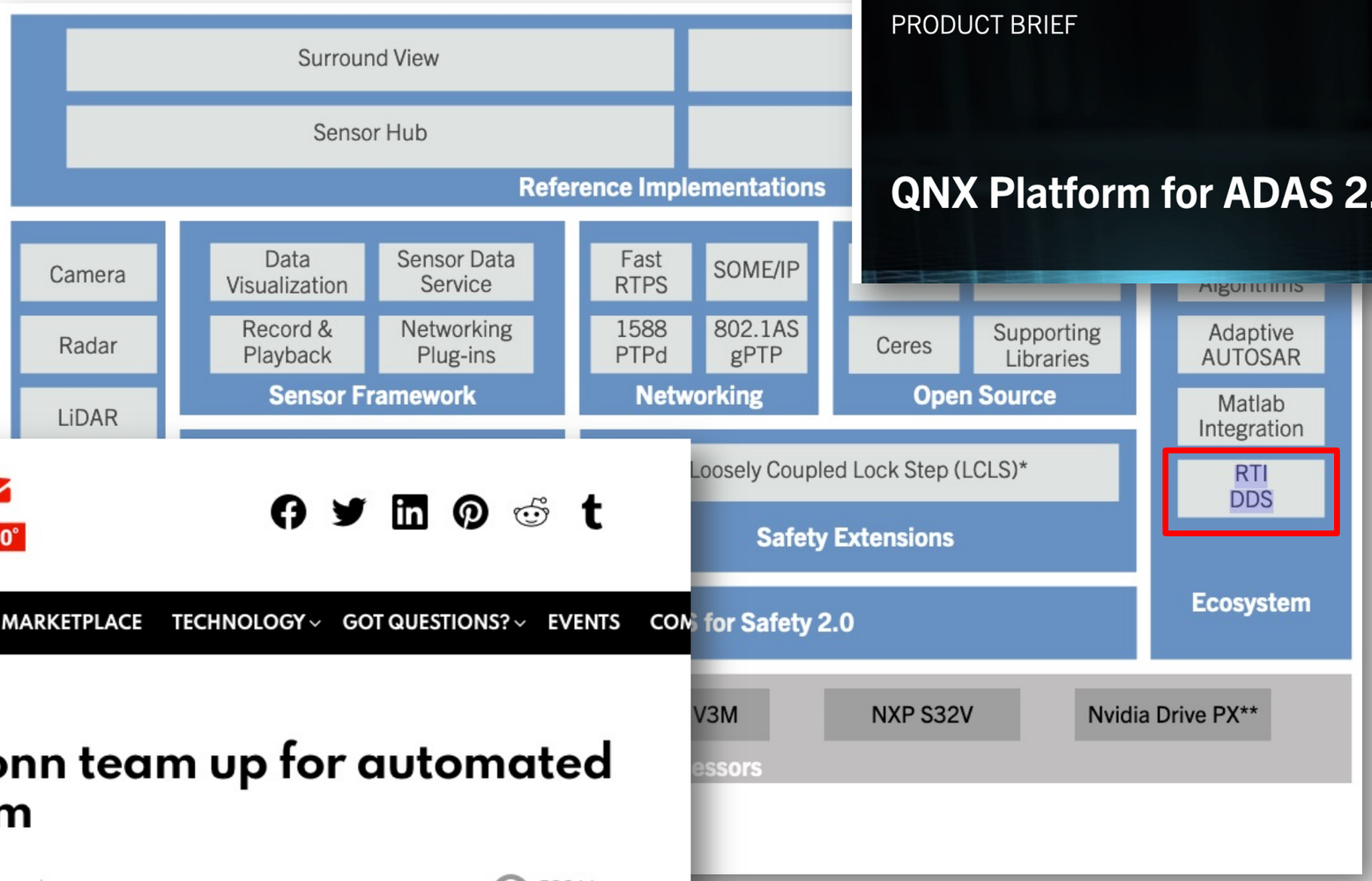
Volkswagen Smart Cars

Volkswagen's Driver Assistance and Integrated Safe
radars, laser range finders, and video to assist safe
eyes, shown here, to detect drowsiness. It also dete
collisions, an

BlackBerry | QNX

PRODUCT BRIEF

QNX Platform for ADAS 2.0



SELF-DRIVING CARS 360°
INDUSTRY NEWS AND COMMUNITY

NEWS COMPANIES MARKETPLACE TECHNOLOGY GOT QUESTIONS? EVENTS COM

Industry news & trends

Adlink, Foxconn team up for automated driving system

by Moderation Team · 5 months ago

598 Views

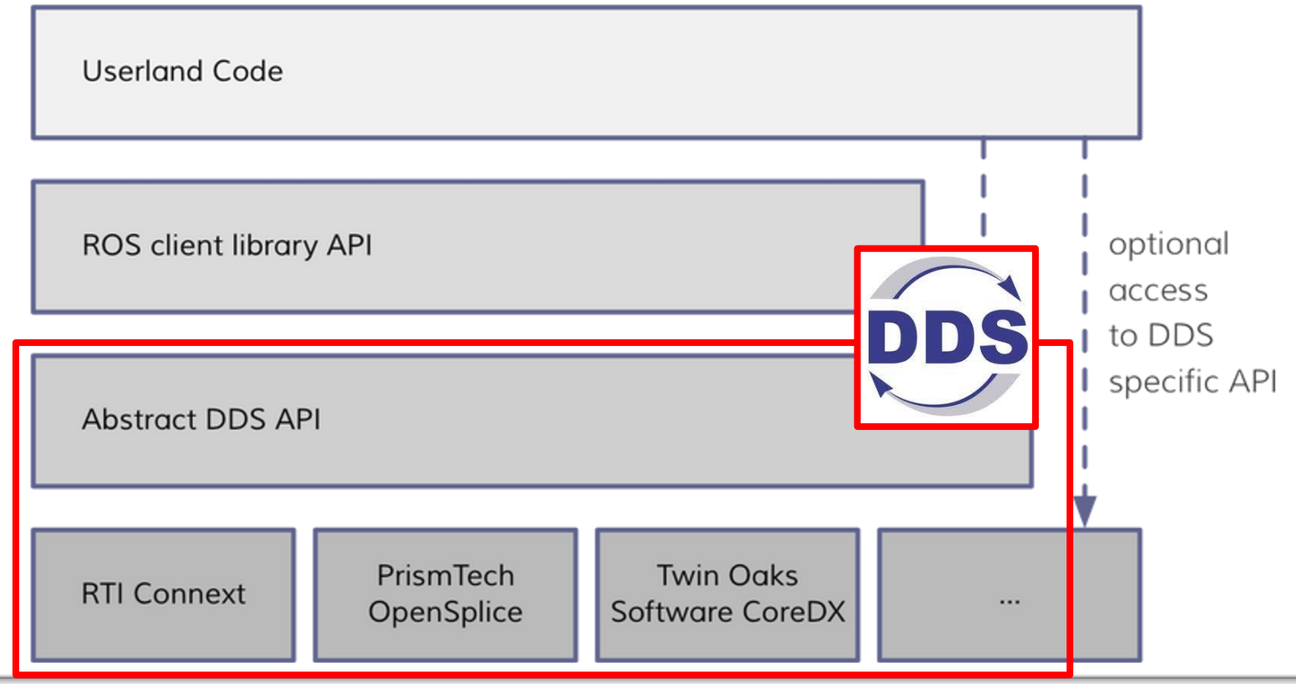
17 SHARES

Share Tweet

Next-gen Robotics

ROS Built on DDS

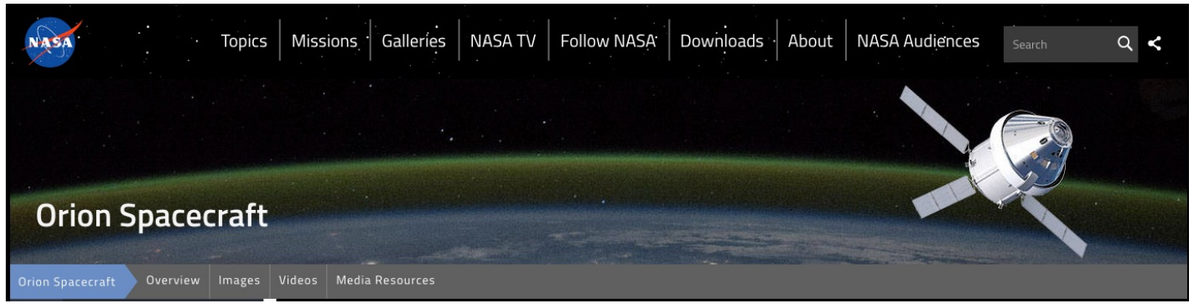
The goal is to make DDS an implementation detail of ROS 2. This means that all DDS specific APIs and message definitions would need to be hidden. DDS provides discovery, message definition, message serialization, and publish-subscribe transport. Therefore, DDS would provide discovery, publish-subscribe transport, and at least the underlying message serialization for ROS. ROS 2 would provide a ROS 1 like interface on top of DDS which hides much of the complexity of DDS for the majority of ROS users, but then separately provides access to the underlying DDS implementation for users that have extreme use cases or need to integrate with other, existing DDS systems.



https://design.ros2.org/articles/ros_on_dds.html

Aerospace

https://archive.is/wip/Sjjsq



https://archive.is/wip/MNhkB https://software.nasa.gov/software/KSC-13925

Healthcare

GE Healthcare

Log in or register to post comments Last post

Sat, 05/08/2021 - 02:05

anotherChris
Offline
Last seen: 1 month 1 week ago
Joined: 05/08/2021
Posts: 2

What means this error message: ModuleID=6 Errcode=59

Hello,

This is with DDS Micro 2.4.11

I have this following error message:

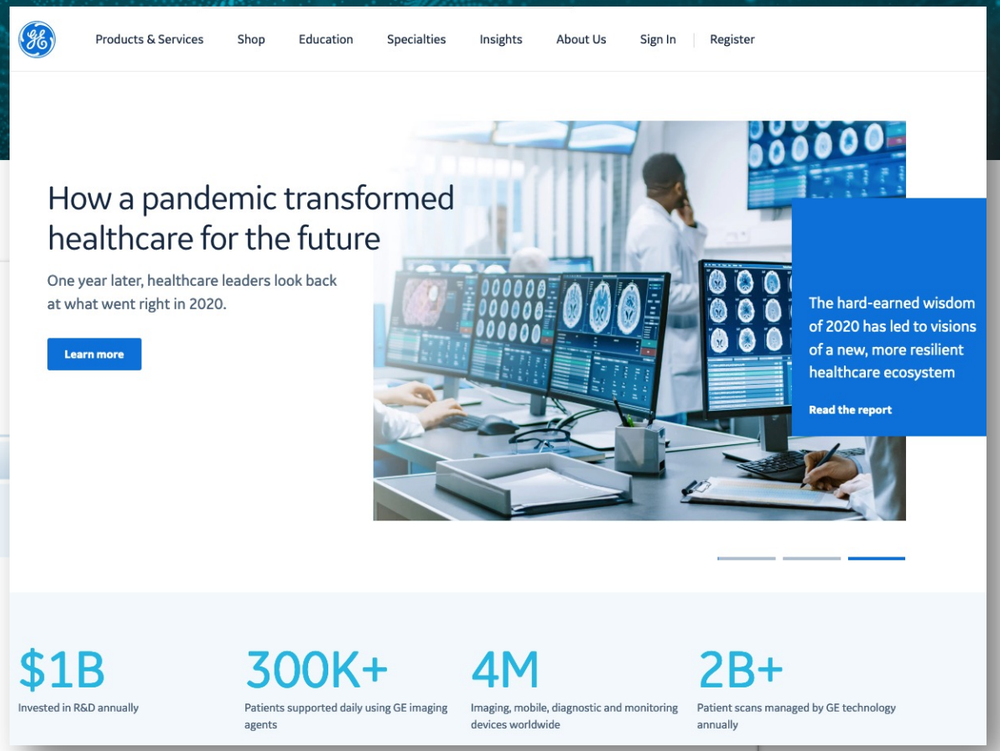
[15.700000000]ERROR: ModuleID=6 Errcode=59 X=1 E=0 T=1 undefined/RTPSInterface.c:6927/RTPS_Interface_receive

I am not able to interpret what it means, can anybody explain the root cause and how to solve it ?

Thank you,

Chris

Organization:
GE Healthcare



https://community.rti.com/organization/ge-healthcare

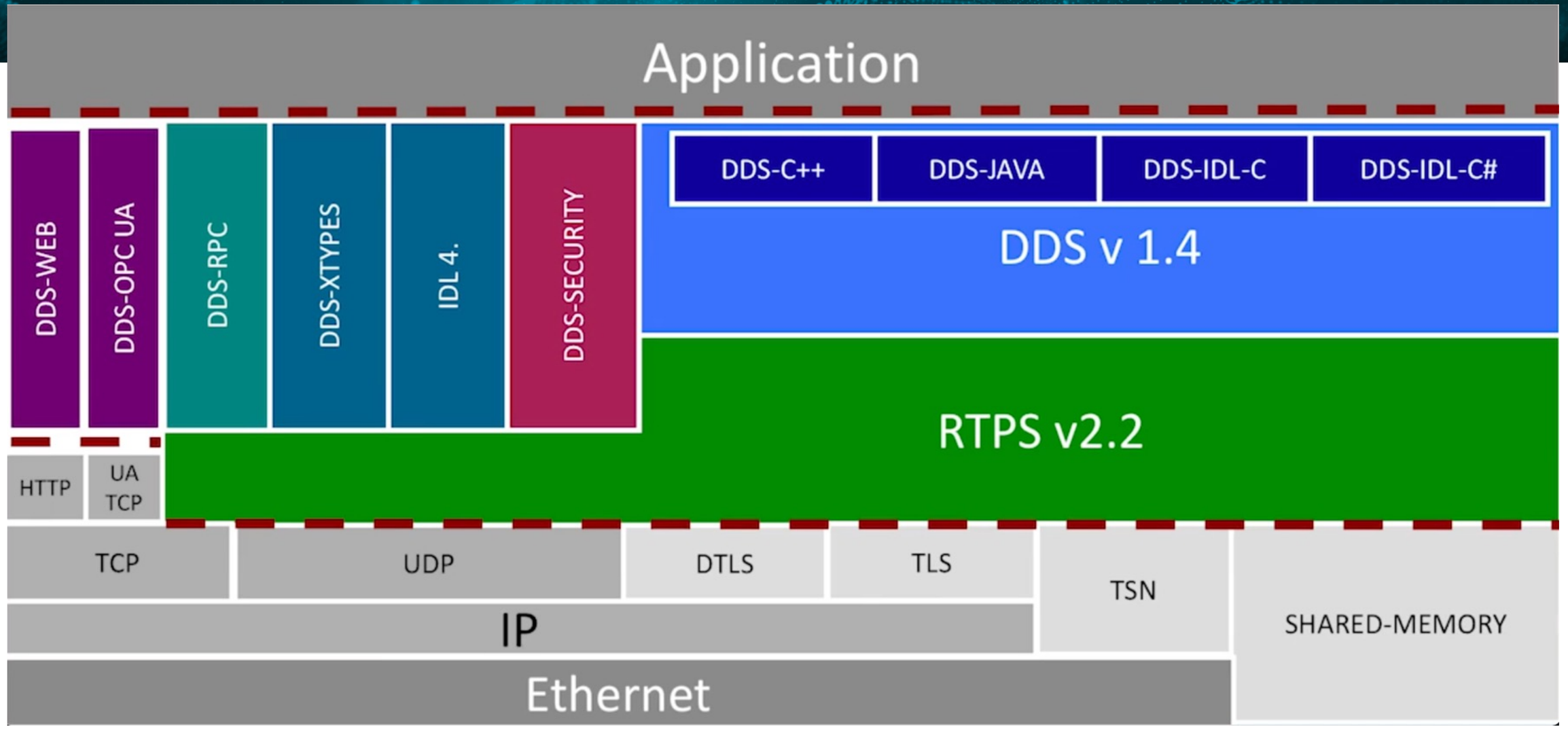
black hat EUROPE 2021

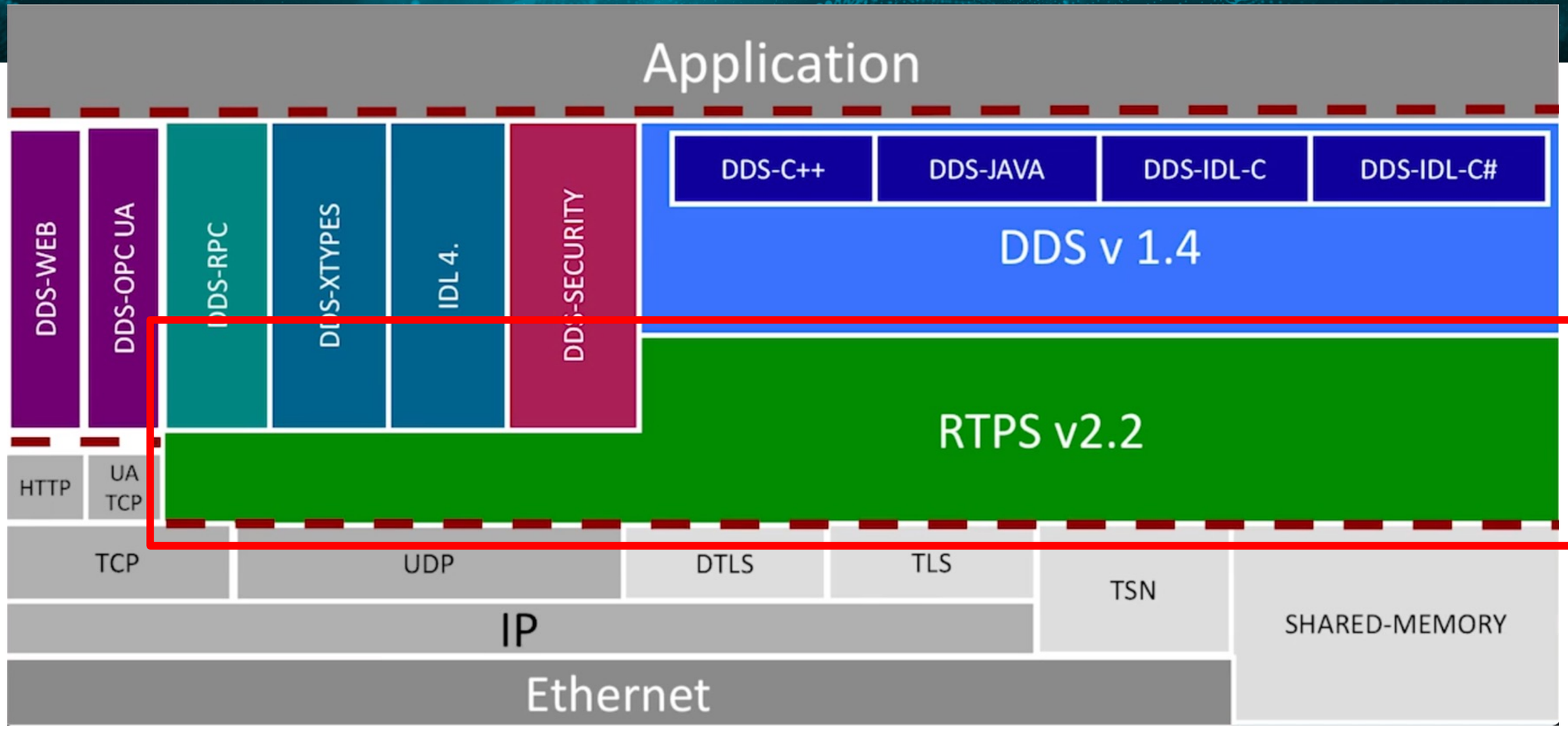
november 10-11, 2021

BRIEFINGS

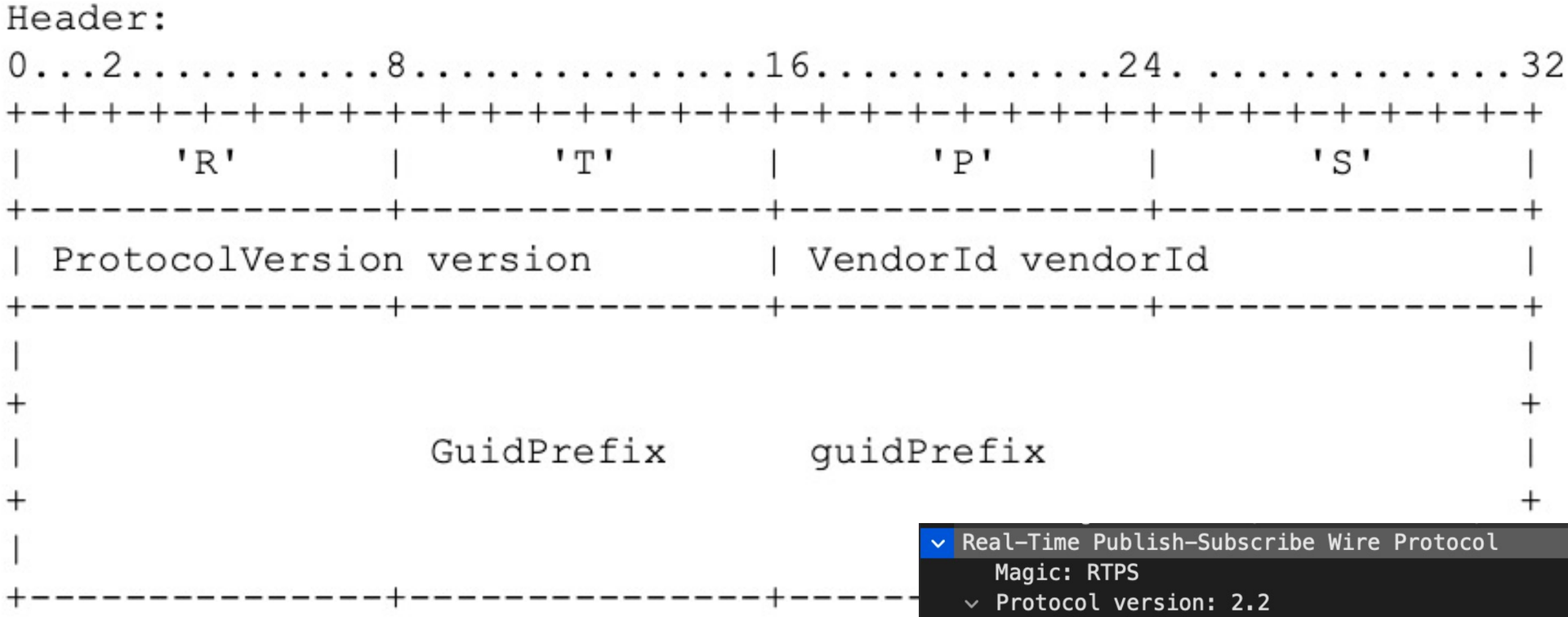
Let's dive into DDS

Federico Maggi Trend Micro Research


Application
DDS-WEB
DDS-OPC UA
DDS-RPC
DDS-XTYPES
IDL 4.
DDS-SECURITY
DDS-C++
DDS-JAVA
DDS-IDL-C
DDS-IDL-C#
DDS v 1.4
RTPS v2.2
HTTP
UA
TCP
TCP
UDP
DTLS
TLS
TSN
SHARED-MEMORY
IP
Ethernet

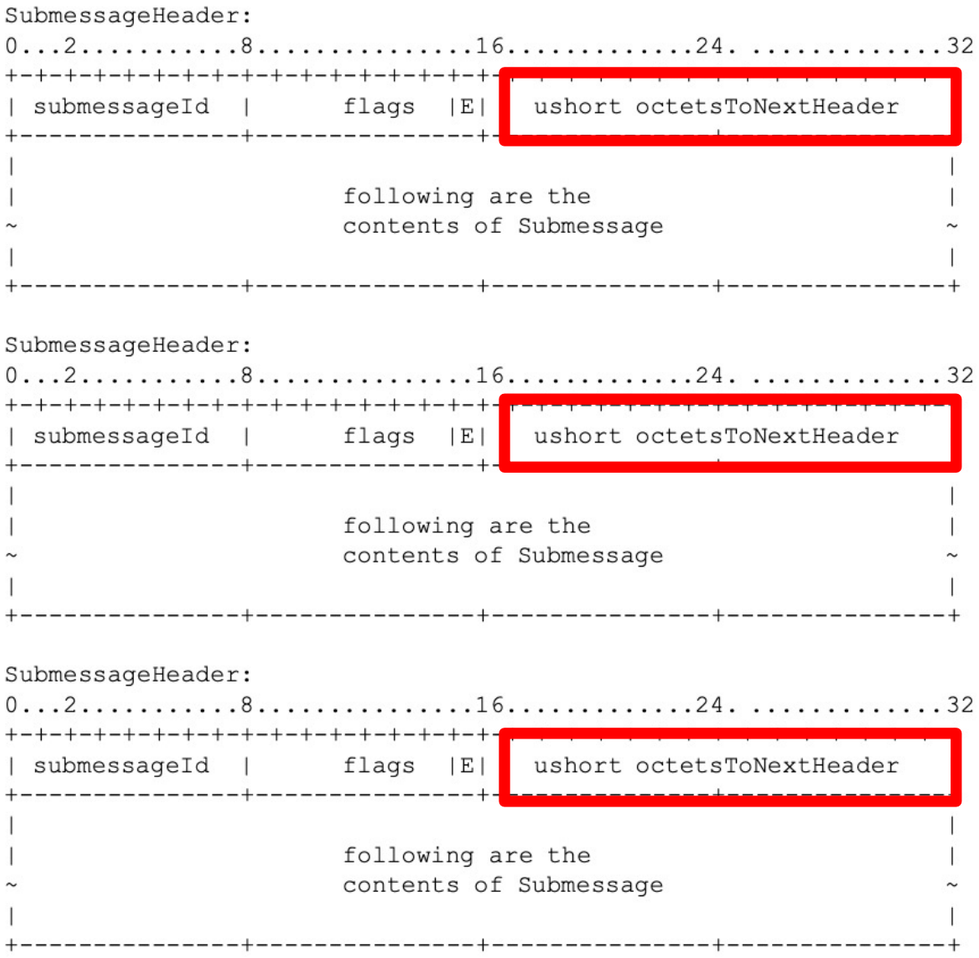

Application
DDS-WEB
DDS-OPC UA
DDS-RPC
DDS-XTYPES
IDL 4.
DDS-SECURITY
DDS-C++
DDS-JAVA
DDS-IDL-C
DDS-IDL-C#
DDS v 1.4
RTPS v2.2
HTTP
UA
TCP
TCP
UDP
DTLS
TLS
TSN
SHARED-MEMORY
IP
Ethernet

RTPS Header

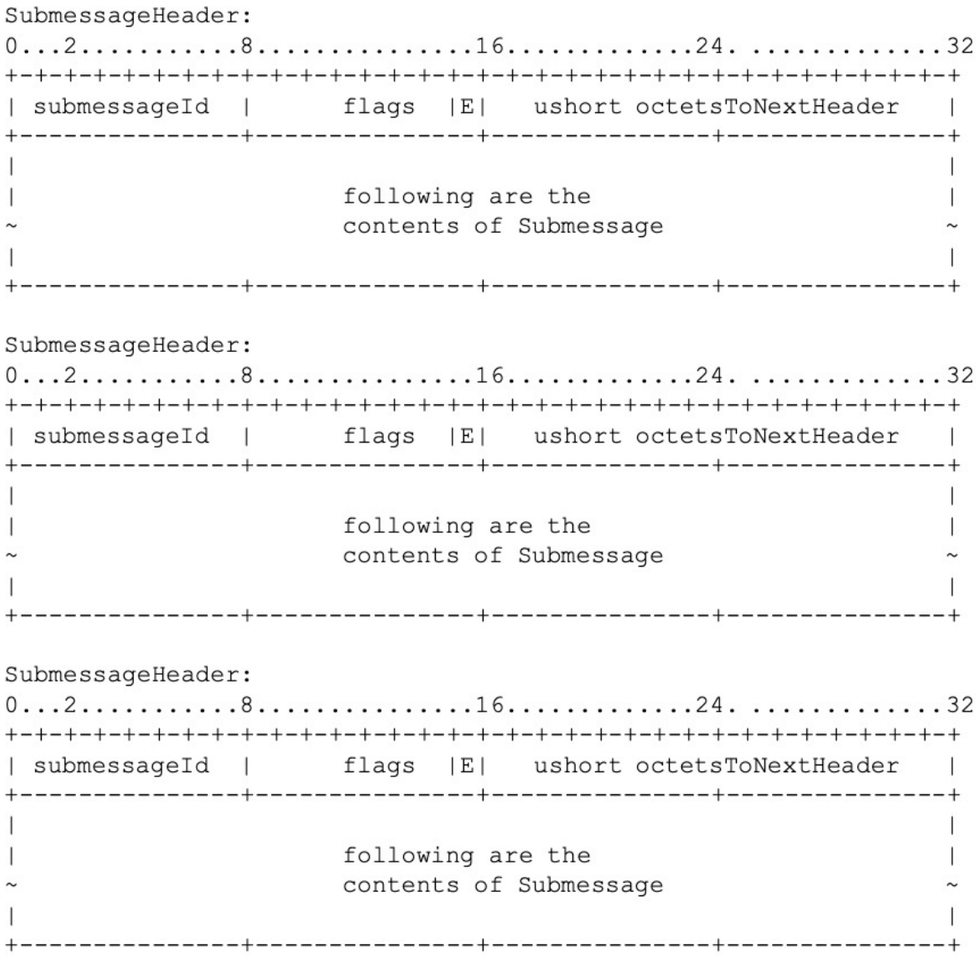



```
Real-Time Publish-Subscribe Wire Protocol
  Magic: RTPS
  Protocol version: 2.2
    major: 2
    minor: 2
  vendorId: 01.15 (eProsima - Fast-RTPS)
  guidPrefix: 010f7f014c0d00000a000000
    hostId: 0x010f7f01
    appId: 0x4c0d0000
    instanceId: 0x0a000000
  Default port mapping: domainId=2, participantIdx=8, nature=UNICAST_METATRAFFIC
    [domain_id: 2]
    [participant_idx: 8]
    [traffic_nature: UNICAST_METATRAFFIC (0)]
```

RTPS: list of sub-messages

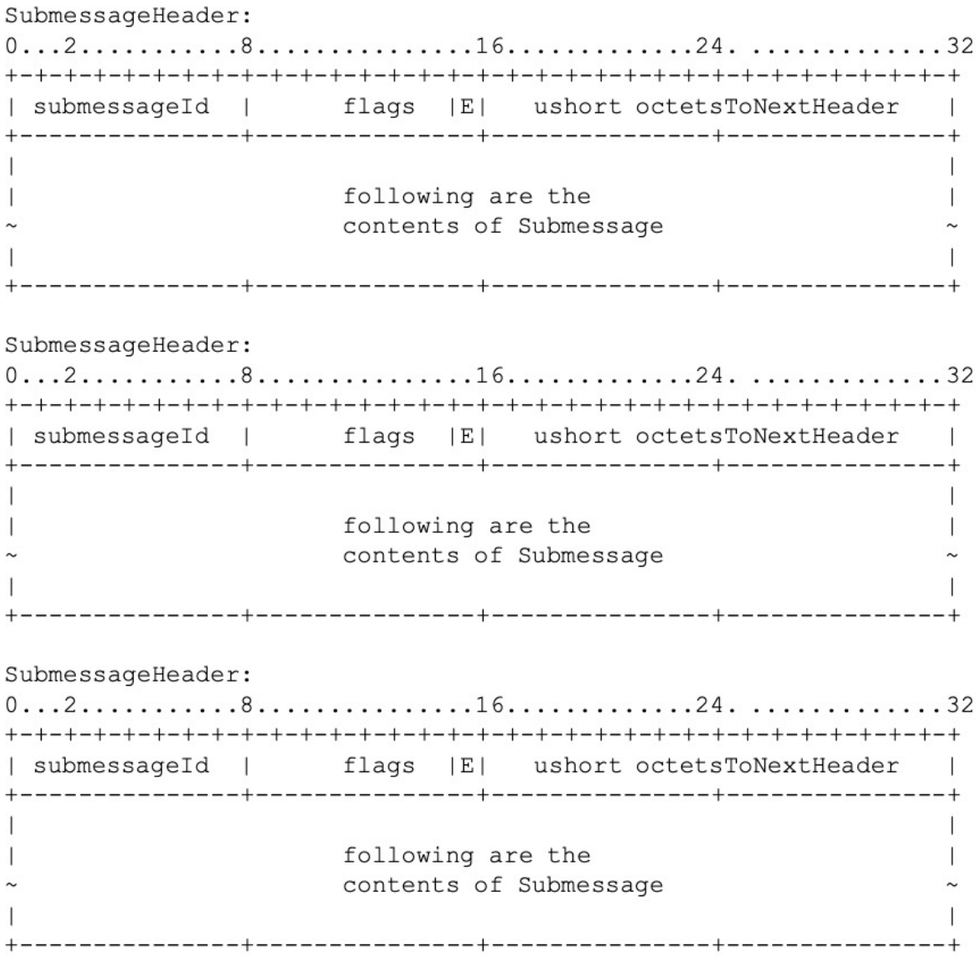


RTPS sub-message kind



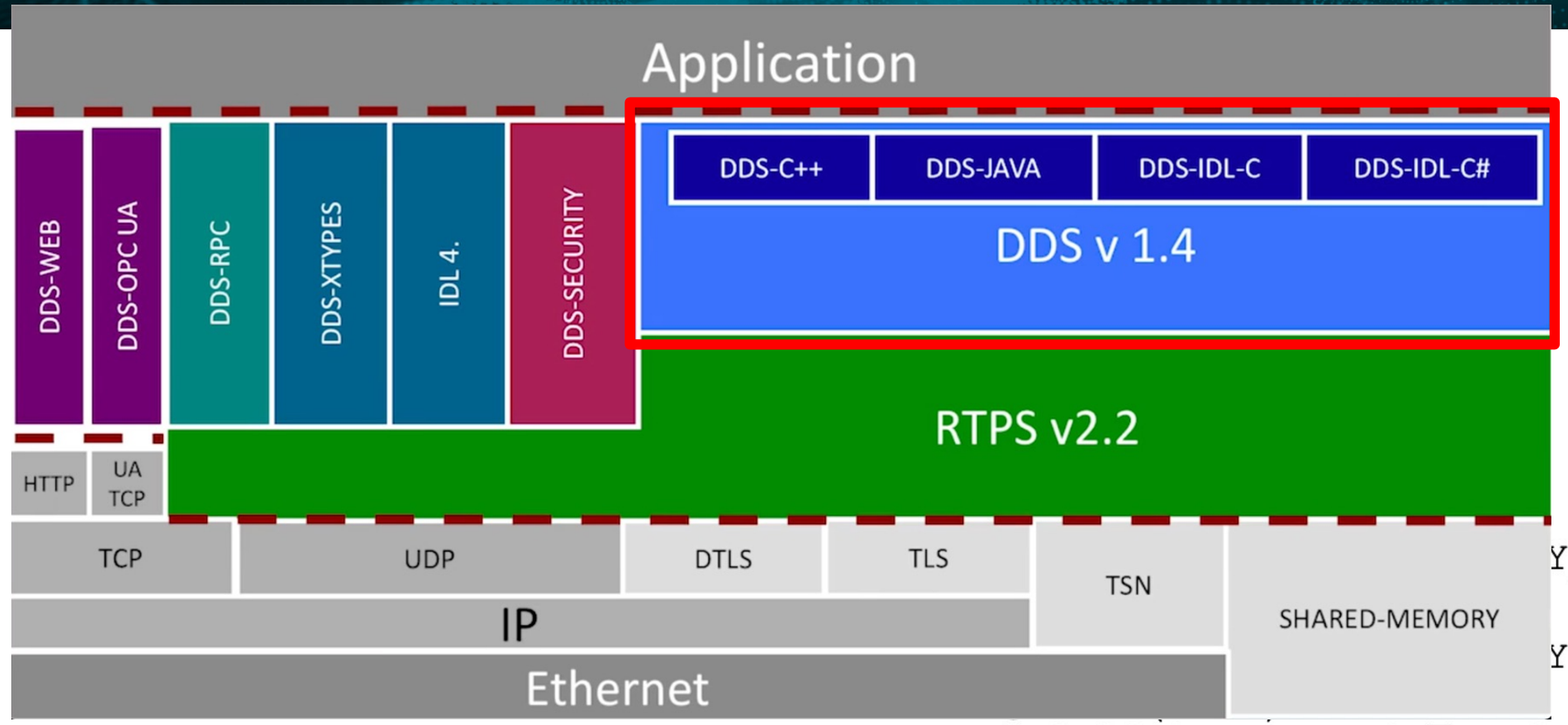

```
enum SubmessageKind {
    @value(0x00)   RTPS_HE,          /* HeaderExtension */
    @value(0x01)   PAD,              /* Pad */
    @value(0x06)   ACKNACK           /* AckNack */
    @value(0x07)   HEARTBEAT         /* Heartbeat */
    @value(8x08)   GAP               /* Gap */
    @value(0x09)   INFO_TS           /* InfoTimestamp */
    @value(0x0c)   INFO_SRC          /* InfoSource */
    @value(0x0d)   INFO_REPLY_IP4    /* InfoReplyIp4 */
    @value(0x0e)   INFO_DST          /* InfoDestination */
    @value(0x0f)   INFO_REPLY        /* InfoReply */
    @value(0x12)   NACK_FRAG         /* NackFrag */
    @value(0x13)   HEARTBEAT_FRAG    /* HeartbeatFrag */
    @value(0x15)   DATA              /* Data */
    @value(0x16)   DATA_FRAG         /* DataFrag */
};
```

RTPS sub-messages: INFO_TS (0x09)




```
enum SubmessageKind {
    @value(0x00)  RTPS_HE,          /* HeaderExtension */
    @value(0x01)  PAD,              /* Pad */
    @value(0x06)  ACKNACK           /* AckNack */
    @value(0x07)  HEARTBEAT         /* Heartbeat */
    @value(8x08)  GAP               /* Gap */
    @value(0x09)  INFO_TS           /* InfoTimestamp */
    @value(0x0c)  INFO_SRC          /* InfoSource */
    @value(0x0d)  INFO_REPLY_IP4    /* InfoReplyIp4 */
    @value(0x0e)  INFO_DST          /* InfoDestination */
    @value(0x0f)  INFO_REPLY        /* InfoReply */
    @value(0x12)  NACK_FRAG         /* NackFrag */
    @value(0x13)  HEARTBEAT_FRAG    /* HeartbeatFrag */
    @value(0x15)  DATA              /* Data */
    @value(0x16)  DATA_FRAG         /* DataFrag */
};
```

RTPS sub-messages: DATA (0x15)

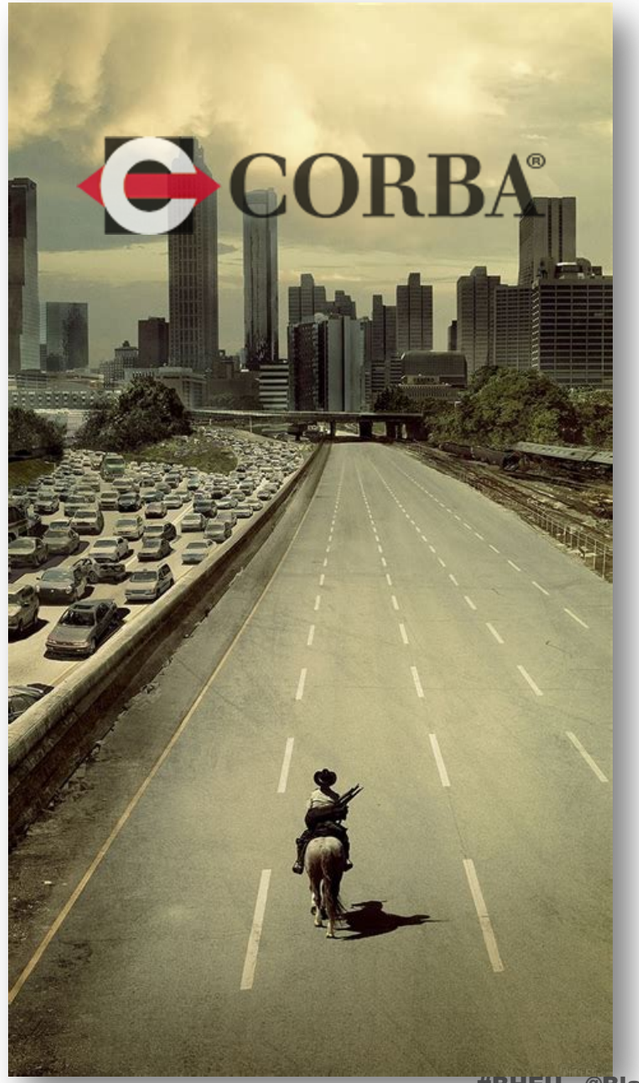



```
                                                    Y_IP4
                                                    Y
        @value(0x13)    HEARTBEAT_FRAG
        @value(0x15)    DATA
        @value(0x16)    DATA_FRAG
};
```

```
/* HeaderExtension */
/* Pad */
/* AckNack */
/* Heartbeat */
/* Gap */
/* InfoTimestamp */
/* InfoSource */
/* InfoReplyIp4 */
/* InfoDestination */
/* InfoReply */
/* NackFrag */
/* HeartbeatFrag */
/* Data */
/* DataFrag */
```

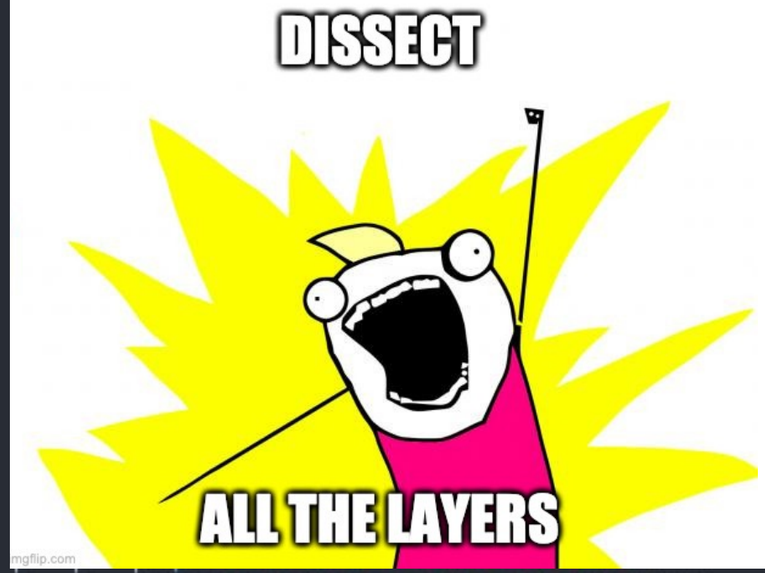
Data present flag: CDR encoding

```
submessageId: DATA (0x15)
  Flags: 0x05, Data present, Endianness bit
    0... .... = Reserved: Not set
    .0.. .... = Reserved: Not set
    ..0. .... = Reserved: Not set
    ...0 .... = Reserved: Not set
    .... 0... = Serialized Key: Not set
    .... .1.. = Data present: Set
    .... ..0. = Inline QoS: Not set
    .... ...1 = Endianness bit: Set
  octetsToNextHeader: 48
  0000 0000 0000 0000 = Extra flags: 0x0000
  Octets to inline QoS: 16
  readerEntityId: 0x00000107 (Application-defined reader (with key): 0x000001)
    readerEntityKey: 0x000001
    readerEntityKind: Application-defined reader (with key) (0x07)
  writerEntityId: 0x00000202 (Application-defined writer (with key): 0x000002)
    writerEntityKey: 0x000002
    writerEntityKind: Application-defined writer (with key) (0x02)
  [Topic Information (from Discovery)]
    [topic: Square]
    [typeName: ShapeType]
    [DCPSPublicationData In: 38]
  writerSeqNumber: 298
  serializedData
    encapsulation kind: CDR_LE (0x0001)
    encapsulation options: 0x0000
    serializedData: 070000004f52414e474500008d0000002d0000001e000000
```



Let’s dissect all the layers

```
###[ RTPS Header ]###
  magic     = 'RTPS'
  \protocolVersion\
   |###[ RTPS Protocol Version ]###
   |  major     = 2
   |  minor     = 1
  \vendorId  \
   |###[ RTPS Vendor ID ]###
   |  vendor_id = b'\x01\x10'
  \guidPrefix\
   |###[ RTPS GUID Prefix ]###
   |  hostId    = 0x57631001
   |  appId     = 0xd6ab407f
   |  instanceId= 0x5bd9bb1c
###[ RTPS Message ]###
     \submessages\
      |###[ RTPS INFO_DTS (0x0e) ]###
      |  submessageId= 0xe
      |  submessageFlags= 0x1
      |  octetsToNextHeader= 12
      |  \guidPrefix\
      |   |###[ RTPS GUID Prefix ]###
      |   |  hostId    = 0x882a1001
      |   |  appId     = 0x5d8c9740
      |   |  instanceId= 0x78b62dc2
      |###[ RTPS INFO_TS (0x09) ]###
      |  submessageId= 0x9
      |  submessageFlags= E
      |  octetsToNextHeader= 8
      |  ts_seconds= 1619087604
      |  ts_fraction= 475848017
```




```
      |###[ PID_DEFAULT_UNICAST_LOCATOR ]###
      |  parameterId= 0x31
      |  parameterLength= 24
      |  \locator   \
      |   |###[ RTPS Locator ]###
      |   |  locatorKind= 0x1000000
      |   |  port      = 60349
      |   |  address   = 172.17.0.2
      |###[ PID_DEFAULT_MULTICAST_LOCATOR ]###
      |  parameterId= 0x48
      |  parameterLength= 24
      |  \locator   \
      |   |###[ RTPS Locator ]###
      |   |  locatorKind= 0x1000000
      |   |  port      = 7401
      |   |  address   = 239.255.0.1
```

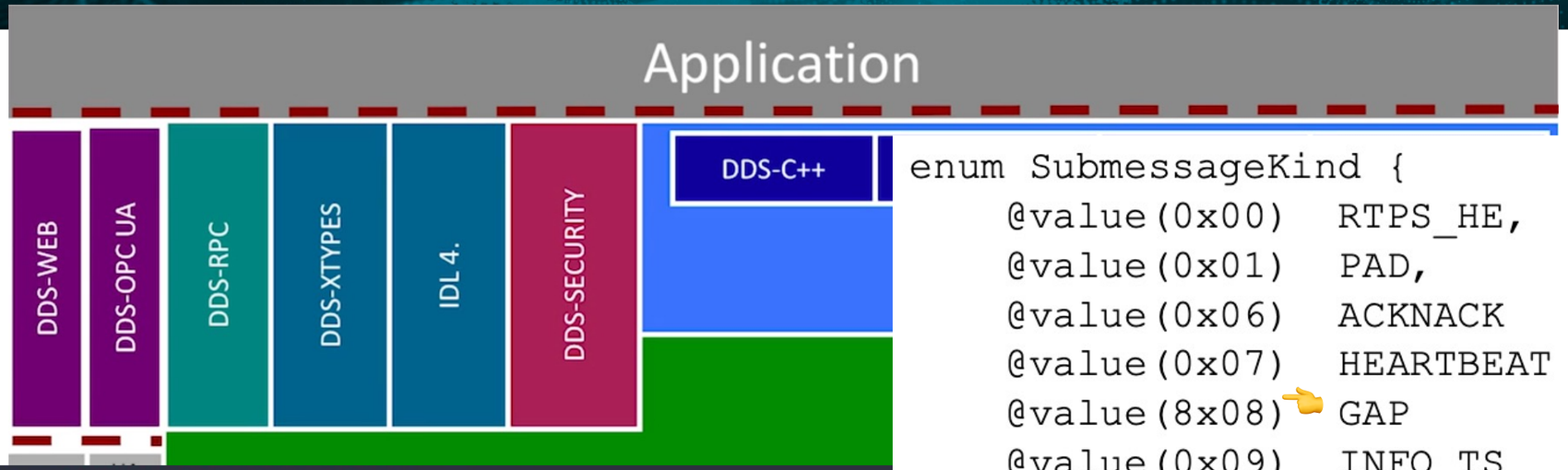
Endianess likes to change

```
submessageId: INFO_TS (0x09)
  Flags: 0x01, Endianness bit
      0... .... = Reserved: Not set
      .0.. .... = Reserved: Not set
      ..0. .... = Reserved: Not set
      ...0 .... = Reserved: Not set
      .... 0... = Reserved: Not set
      .... .0.. = Reserved: Not set
      .... ..0. = Timestamp flag: Not set
      .... ...1 = Endianness bit: Set
    octetsToNextHeader: 8
    Timestamp: Jan 19, 2021 11:48:09.070174100 UTC
```

Dissecting RTPS sub-messages

```
class RTPSMessage(Packet):
    name = "RTPS Message"
    fields_desc = [
        PacketListField("submessages", [],
                        next_cls_cb=lambda pkt, *
                        args: pkt._next_cls_cb(pkt, *args)
                        )
    ]

    def _next_cls_cb(
        self, pkt: Packet, lst: List[Packet], p: Optional[Packet], remain: str
    ) -> Optional[Packet_metaclass]:
        sm_id = struct.unpack("!b", remain[0:1])[0]

        next_cls = _RTPSSubMessageTypes.get(sm_id, None)

        return next_cls
```

Parsing Messages in Scapy: DATA (0x15)




```
enum SubmessageKind {
    @value(0x00)   RTPS_HE,          /* HeaderExtension */
    @value(0x01)   PAD,              /* Pad */
    @value(0x06)   ACKNACK           /* AckNack */
    @value(0x07)   HEARTBEAT         /* Heartbeat */
    @value(8x08)   GAP               /* Gap */
    @value(0x09)   INFO_TS           /* InfoTimestamp */
    @value(0x0c)   INFO_SRC          /* InfoSource */
    @value(0x0d)   INFO_REPLY_IP4    /* InfoReplyIp4 */
    @value(0x0e)   INFO_DST          /* InfoDestination */
    @value(0x0f)   INFO_REPLY        /* InfoReply */
    @value(0x12)   NACK_FRAG         /* NackFrag */
    @value(0x13)   HEARTBEAT_FRAG    /* HeartbeatFrag */
    @value(0x15)   DATA              /* Data */
    @value(0x16)   DATA_FRAG         /* DataFrag */
};
```

```
class RTPSSubMessage_DATA(EPacket):
    name = "RTPS DATA (0x15)"
    fields_desc = [
        XByteField("submessageId", 0),
        XByteField("submessageFlags", 0x00),
        EField(ShortField("octetsToNextHeader", 0)),
        XNBytesField("extraFlags", 0x0000, 2),
        EField(ShortField("octetsToInlineQoS", 0)),
        X3BytesField("readerEntityIdKey", 0),
        XByteField("readerEntityIdKind", 0),
        X3BytesField("writerEntityIdKey", 0),
        XByteField("writerEntityIdKind", 0),
        EField(IntField("writerSeqNumHi", 0)),
        EField(IntField("writerSeqNumLow", 0)),
        # ------------------------------------
        ConditionalField(
            InlineQoSPacketField("inlineQoS", "", InlineQoSPacket), lambda pkt:pkt.submessageFlags & 0b00000010 == 0b00000010),
        ConditionalField(
            DataPacketField("key", "", DataPacket), lambda pkt:pkt.submessageFlags & 0b00001000 == 0b00001000),
        ConditionalField(
            DataPacketField("data", "", DataPacket), lambda pkt:pkt.submessageFlags & 0b00000100 == 0b00000100),
    ]
```

DataPacket (and handling endianess)

```
class DataPacket(EPacket):
    name = "Data Packet"
    _pl_type = None
    _pl_len = 0

    fields_desc = [
        XShortField("encapsulationKind", 0),
        XShortField("encapsulationOptions", 0),
        MultipleTypeField([
            # if payload encoding == PL_CDR_{LE,BE} then parameter list
            (EPacketField("parameterList", "", ParameterListPacket),
             lambda pkt:pkt.encapsulationKind == 0x0003),
            # else if writer entity id == 0x200c2: then participant message data
            (EPacketField("participantMessageData", "", ParticipantMessageDataPacket),
             lambda pkt:pkt._pl_type == "ParticipantMessageData")
        ],
            # else (default)
            SerializedDataField("serializedData", "", length_from=lambda pkt:pkt._pl_len)
        ),
    ]
```

Grab the code!

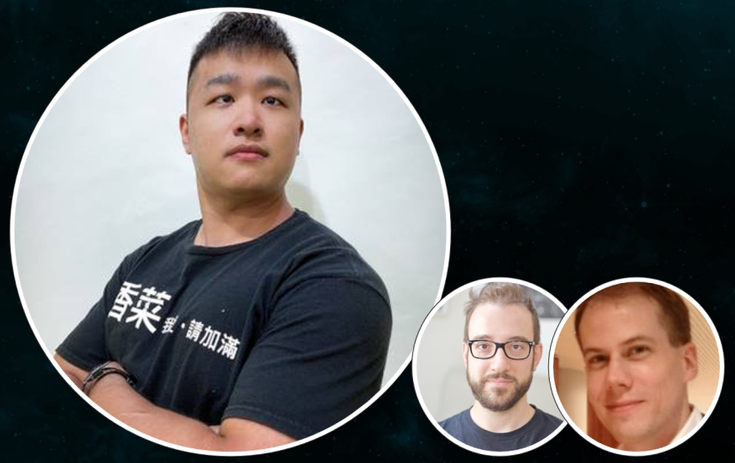
```
class ParameterListPacket(EPacket):
    name = "PID list"
    fields_desc = [
        PacketListField("parameterValues", [], next_cls_cb=get_pid_class),
        PacketField("sentinel", "", PID_SENTINEL)
    ]
```

https://github.com/secdev/scapy/pull/3403

```
class PID_PROTOCOL_VERSION(PIDPacketBase):
    name = "PID_PROTOCOL_VERSION"
    fields_desc = [
        EField(ParameterIdField("parameterId", 0)),
        EField(ShortField("parameterLength", 0)),
        PacketField("protocolVersion", "", ProtocolVer
        StrLenField(
            "padding", "", length_from=lambda x:x.para
    ]


class PID_VENDOR_ID(PIDPacketBase):
    name = "PID_VENDOR_ID"
    fields_desc = [
        EField(ParameterIdField("parameterId", 0)),
        EField(ShortField("parameterLength", 0)),
        PacketField("vendorId", "", VendorIdPacket),
        StrLenField(
            "padding", "", length_from=lambda x:x.para
    ]


class PID_UNICAST_LOCATOR(PIDPacketBase):
    name = "PID_UNICAST_LOCATOR"
    fields_desc = [
        EField(ParameterIdField("parameterId", 0)),
        EField(ShortField("parameterLength", 0)),
        PacketField("locator", "", LocatorPacket)
    ]


class PID_MULTICAST_LOCATOR(PIDPacketBase):
    name = "PID_MULTICAST_LOCATOR"
    fields_desc = [
        EField(ParameterIdField("parameterId", 0)),
        EField(ShortField("parameterLength", 0)),
        StrLenField("parameterData", "",
                    length_from=lambda x: x.parameterL
    ]
```

black hat EUROPE 2021
november 10-11, 2021
BRIEFINGS

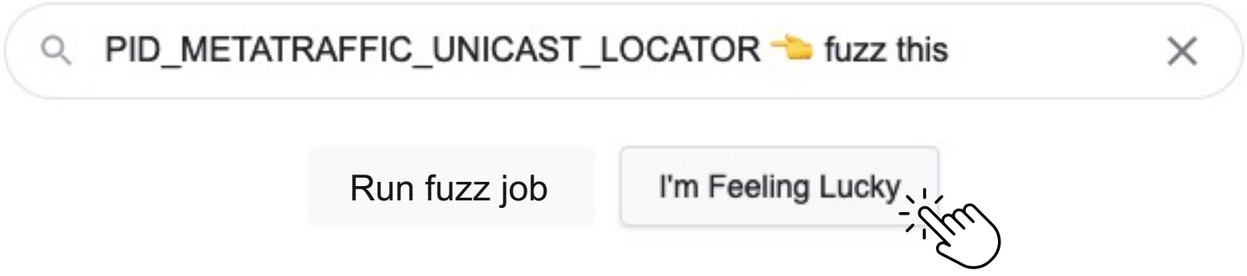
Let's fuzz all the things



Ta-Lun Yen TXOne Networks

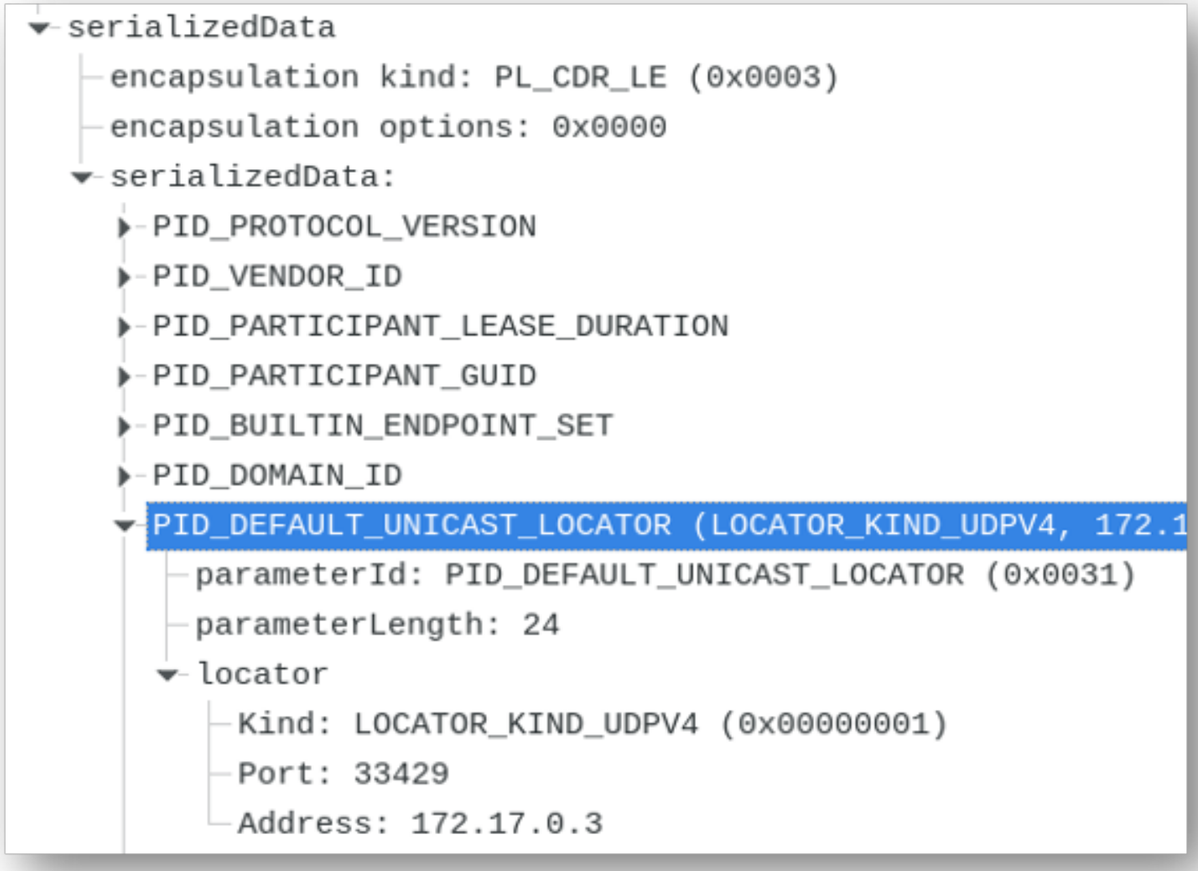
Smartest Fuzzing ==

When you know the protocol so well that you can stick a needle in the field you feel it's the right one



Fumbling with fields manually

- Change…
  - ID to non-existing parameterID
  - Len/port to 0xFF…FF/0x00
  - IP to some random address



PID_METATRAFFIC_UNICAST_LOCATOR

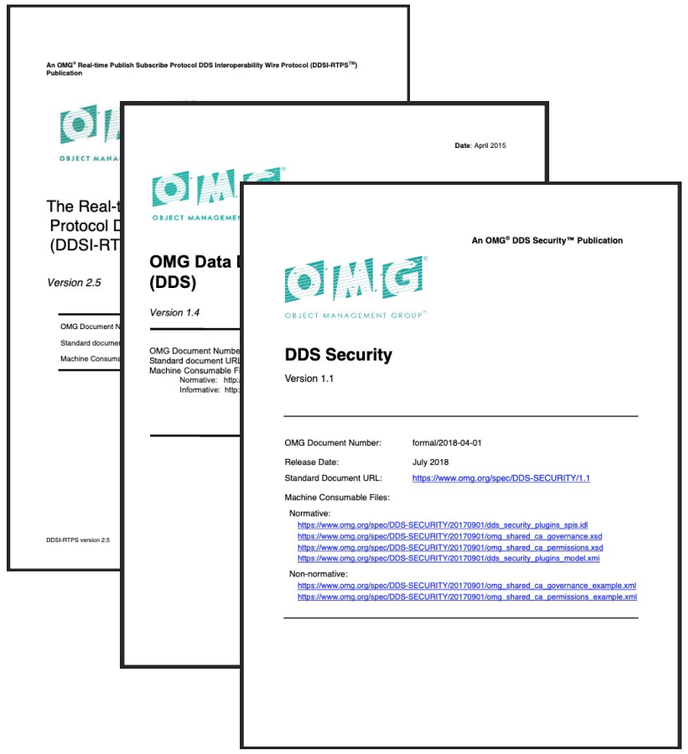
- Grab a valid RTPS payload into a byte array

- Change IP to random address & run

172.17.0.4	172.17.0.3	RTPS	350	INFO_TS, DATA(p)
172.17.0.3	8.8.8.8	RTPS	366	INFO_DST, INFO_TS, DATA(p)
172.17.0.3	8.8.8.8	RTPS	174	INFO_DST, HEARTBEAT, HEARTBEAT, HEARTE
172.17.0.3	8.8.8.8	RTPS	162	INFO_DST, ACKNACK, ACKNACK, ACKNACK
172.17.0.3	8.8.8.8	RTPS	174	INFO_DST, HEARTBEAT, HEARTBEAT, HEARTE
172.17.0.3	8.8.8.8	RTPS	174	INFO_DST, HEARTBEAT, HEARTBEAT, HEARTE
172.17.0.3	8.8.8.8	RTPS	174	INFO_DST, HEARTBEAT, HEARTBEAT, HEARTE
172.17.0.3	8.8.8.8	RTPS	174	INFO_DST, HEARTBEAT, HEARTBEAT, HEARTE
172.17.0.1	172.17.[illegible]		86	57621 → 57621 Len=44
172.17.0.3	8.8.8.8		174	INFO_DST, HEARTBEAT, HEARTBEAT, HEARTE
172.17.0.3	8.8.8.8		174	INFO_DST, HEARTBEAT, HEARTBEAT, HEARTE
172.17.0.3	8.8.8.8		174	INFO_DST, HEARTBEAT, HEARTBEAT, HEARTE
172.17.0.3	8.8.8.8		366	INFO_DST, INFO_TS, DATA(p)
172.17.0.3	8.8.8.8		174	INFO_DST, HEARTBEAT, HEARTBEAT, HEARTE
172.17.0.3	8.8.8.8		162	INFO_DST, ACKNACK, ACKNACK, ACKNACK

Reflection vulnerability

- Grab a valid RTPS payload into a byte array
- Change IP to arbitrary address & run
  - Reflection attack (~8–36 BAF)
  - Works on **all implementations**
  - **DDS Specification is broken**
    - **Allows arbitrary IP in the field**
    - **No white-listing**

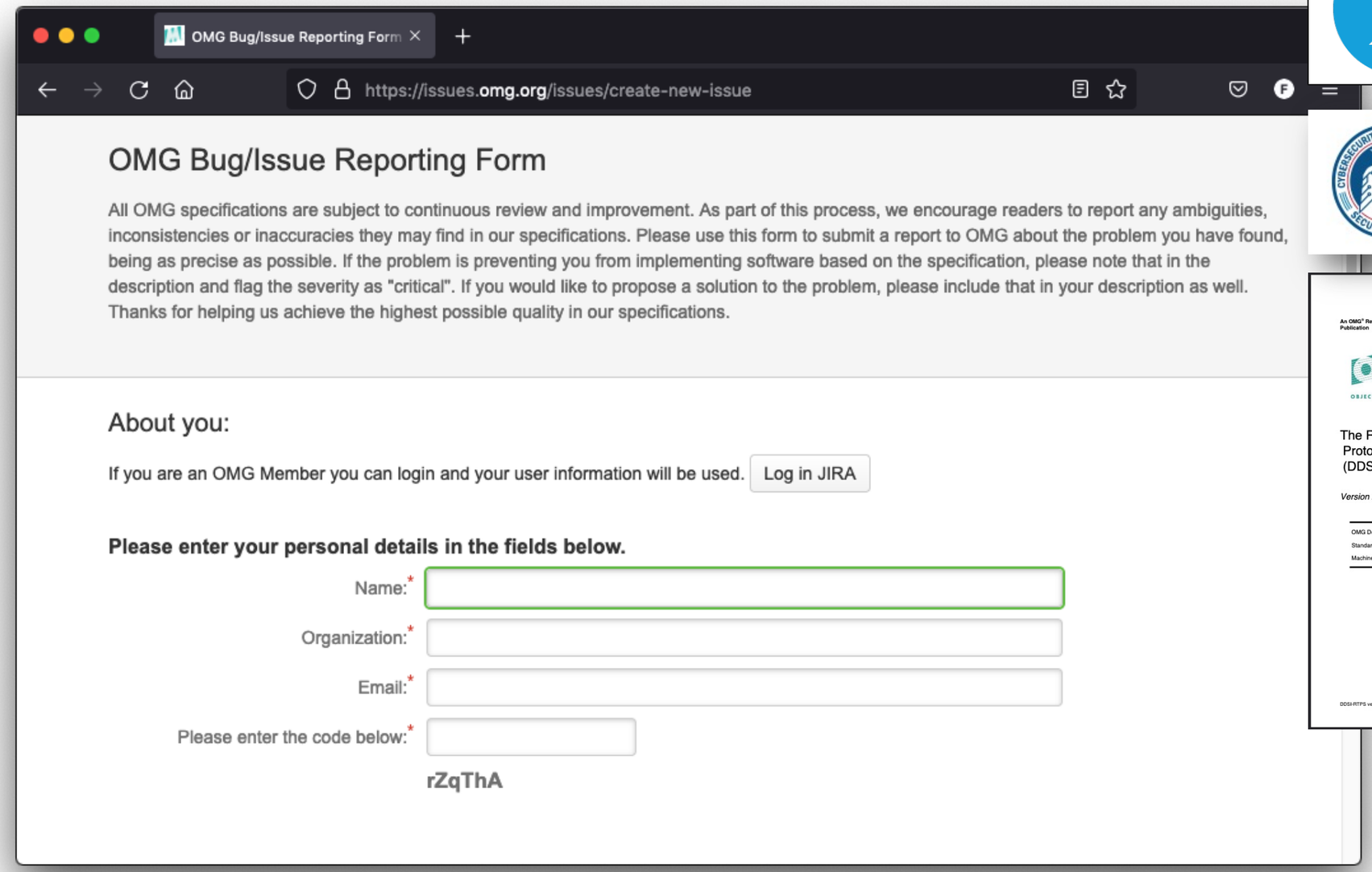


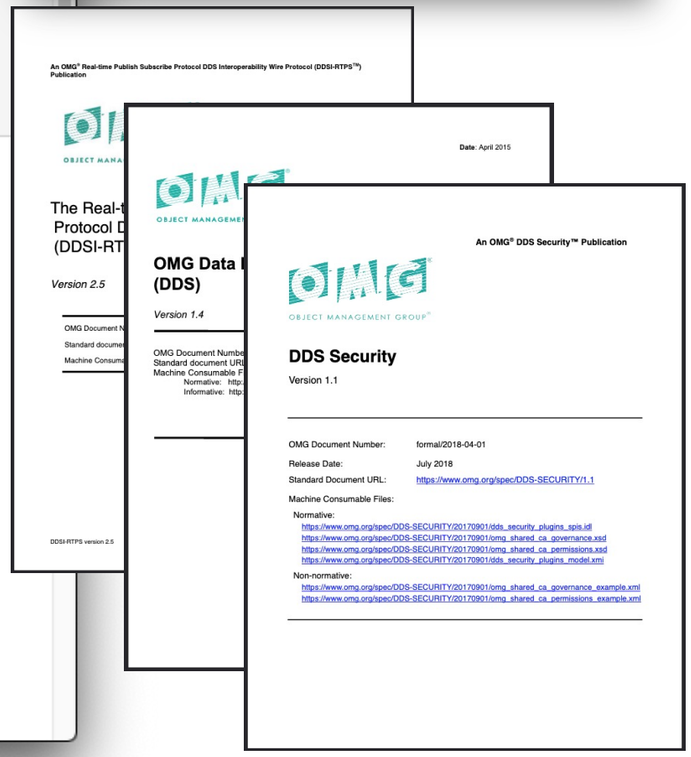
DEMO

RTPS Network Reflection

https://github.com/the-dds/pocs

Disclosing a spec-level vulnerability

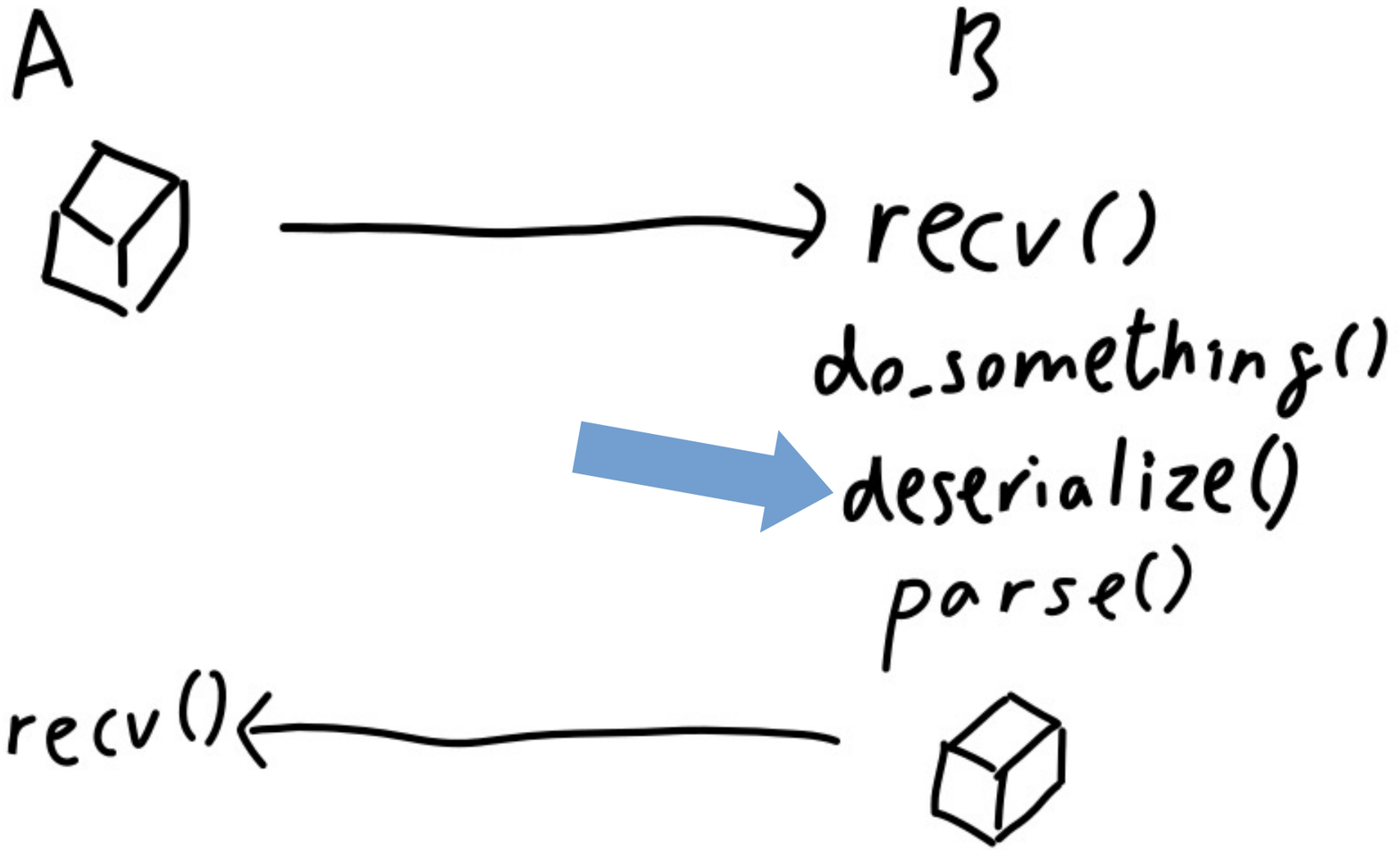




	CVE	CWE	Notes	Status
All	**-**	**CWE-406**	**Network reflection**	
OMG (specs)	**-**	**Extend specs to allow white-listing**		**Disclosed**
RTI ConnextDDS		Patched in the most current release		Patched
OCI OpenDDS	CVE-2021-38429	Patched in the most current release		Mitigated with patch
ADLINK CycloneDDS	-	Had already an exp. back-off mechanism		Already mitigated
GurumDDS	-	Had already an exp. back-off mechanism		Already mitigated (No reply, 5 times)
eProsima Fast-DDS	CVE-2021-38429	Patched in the master branch		Mitigated with patch
Twin Oaks CoreDX	N/A	N/A		Disclosed

Whitebox Application Fuzzing

Thinking outside the box



deserialize()

- Let’s look into the (decompiled) code for:
  - Valid “preamble” (e.g., “RTPS”)
  - Parsing headers (e.g., vendor ID, version)
  - Error messages
    - e.g., “`deserializ* failed`”
  - Switch case for determining PID & corresponding handlers

Quick Overview of Good Fuzz Targets

How to Find Good Fuzz Targets: Serializer

```
(gdb) bt
#0  OpenDDS::DCPS::Serializer::smemcpy (this=0x7ffc615e18a0, to=0x7ffc614dddc0 "w", from=0x612000000078 "\017", n=2) at DCPS/Serializer.cpp:374
#1  0x0000000004d3d1b in OpenDDS::DCPS::Serializer::doread (this=0x7ffc615e18a0, dest=0x7ffc614dddc0 "w", size=2, swap=<optimized out>, offset=0)
#2  OpenDDS::DCPS::Serializer::buffer_read (this=<optimized out>, dest=<optimized out>, size=<optimized out>, swap=<optimized out>) at /usr/local/
#3  0x00007fdfa4d1439b in OpenDDS::DCPS::operator>> (s=..., x=@0x7ffc614dddc0: 119) at DCPS/Serializer.inl:1173
#4  0x00007fdfa16e0d2f in OpenDDS::DCPS::operator>> (outer_strm=..., uni=...) at RtpsCoreTypeSupportImpl.cpp:11440
#5  0x00007fdfa16dff18 in OpenDDS::DCPS::operator>> (strm=..., seq=...) at RtpsCoreTypeSupportImpl.cpp:8985
#6  0x0000000004d051f in main (argc=<optimized out>, argv=<optimized out>) at test.cpp:106
```

```
if ((buff_.size() >= 4) && ACE_OS::memcmp(buff_.rd_ptr(), "RTPS", 4) == 0) {
  RTPS::Message message;

  DCPS::Serializer ser(&buff_, encoding_plain_native);
  Header header;
  if (!(ser >> header)) {
    ACE_ERROR((LM_ERROR,
               ACE_TEXT("(%P|%t) ERROR: Spdp::SpdpTransport::handle_input() - ")
               ACE_TEXT("failed to deserialize RTPS header for SPDP\n")));
    return 0;
  }
```

```
while (buff_.length() > 3) {
  const char subm = buff_.rd_ptr()[0], flags = buff_.rd_ptr()[1];
  ser.swap_bytes((flags & FLAG_E) != ACE_CDR_BYTE_ORDER);
  const size_t start = buff_.length();
  CORBA::UShort submessageLength = 0;
  switch (subm) {
  case DATA: {
```

How to Find Good Fuzz Targets: Parser

```
while (buff_.length() > 3) {
  const char subm = buff_.rd_ptr()[0], flags = buff_.rd_ptr()[1];
  ser.swap_bytes((flags & FLAG_E) != ACE_CDR_BYTE_ORDER);
  const size_t start = buff_.length();
  CORBA::UShort submessageLength = 0;
  switch (subm) {
  case DATA: {
    DataSubmessage data;
    if (!(ser >> data)) {
      ACE_ERROR((LM_ERROR,
                 ACE_TEXT("(%P|%t) ERROR: Spdp::SpdpTransport::handle_input() - ")
                 ACE_TEXT("failed to deserialize DATA header for SPDP\n")));
      return 0;
    }
    submessageLength = data.smHeader.submessageLength;

    if (DCPS::transport_debug.log_messages) {
      append_submessage(message, data);
    }

    if (data.writerId != ENTITYID_SPDP_BUILTIN_PARTICIPANT_WRITER) {
      // Not our message: this could be the same multicast group used
      // for SEDP and other traffic.
      break;
    }

    ParameterList plist;
    if (data.smHeader.flags & (FLAG_D | FLAG_K_IN_DATA)) {
      ser.swap_bytes(!ACE_CDR_BYTE_ORDER); // read "encap" itself in LE
      CORBA::UShort encap, options;
      if (!(ser >> encap) || (encap != encap_LE && encap != encap_BE)) {
        ACE_ERROR((LM_ERROR,
                   ACE_TEXT("(%P|%t) ERROR: Spdp::SpdpTransport::handle_input() - ")
                   ACE_TEXT("failed to deserialize encapsulation header for SPDP\n")));
        return 0;
      }
      ser >> options;
      // bit 8 in encap is on if it's PL_CDR_LE
      ser.swap_bytes(((encap & 0x100) >> 8) != ACE_CDR_BYTE_ORDER);
      if (!(ser >> plist)) {
        ACE_ERROR((LM_ERROR,
                   ACE_TEXT("(%P|%t) ERROR: Spdp::SpdpTransport::handle_input() - ")
                   ACE_TEXT("failed to deserialize data payload for SPDP\n")));
        return 0;
      }
    } else {
      plist.length(1);
      const RepoId guid = make_id(header.guidPrefix, ENTITYID_PARTICIPANT);
      plist[0].guid(guid);
      plist[0]._d(PID_PARTICIPANT_GUID);
    }

    DCPS::RcHandle<Spdp> outer = outer_.lock();

    if (outer) {
      outer->data_received(data, plist, remote);
    }
    break;
```

```
case DATA: {
  DataSubmessage data;
```

```
case INFO_DST: {
  if (DCPS::transport_debug.log_messages) {
    InfoDestinationSubmessage sm;
```

```
case HEARTBEAT:
  if (!check_encoded(submessage.heartbeat_sm().writerId)) {
    if (transport_debug.log_dropped_messages) {
      ACE_DEBUG((LM_DEBUG, "(%P|%t) {transport_debug.log_drop
    }
```

Same Routine, Different Implementation

```
nn_rtps_msg_state_t res = decode_rtps_message (t
if (res != NN_RTPS_MSG_STATE_ERROR)
{
  handle_submsg_sequence (ts1, gv, conn, &srcloc
}
```

```
state_smkind = sm->smhdr.submessageId;
switch (sm->smhdr.submessageId)
{
  case SMID_PAD:
    GVTRACE ("PAD");
    break;
  case SMID_ACKNACK:
    state = "parse:acknack";
    if (!valid_AckNack (rst, &sm->ackna
      goto malformed;
    handle_AckNack (rst, tnowE, &sm->a
    ts_for_latmeas = 0;
    break;
  case SMID_HEARTBEAT:
    state = "parse:heartbeat";
    if (!valid_Heartbeat (&sm->heartbea
      goto malformed;
    handle_Heartbeat (rst, tnowE, rmsg
    ts_for_latmeas = 0;
    break;
  case SMID_GAP:
    state = "parse:gap";
    /* Gap is handled synchronously in
       sometimes have to record a gap
       first case by definition doesn't
       the second one avoids that beca
       rst after inserting the gap in
    if (!valid_Gap (&sm->gap, submsg_s
      goto malformed;
    handle_Gap (rst, tnowE, rmsg, &sm->
    ts_for_latmeas = 0;
    break;
  case SMID_INFO_TS:
```

```
case SMID_DATA:
  state = "parse:data";
  {
    struct nn_rsample_info sampleinfo;
    unsigned char *datap;
    uint32_t datasz = 0;
    size_t submsg_len = submsg_size;
    /* valid_Data does not validate the payload */
    if (!valid_Data (rst, &sm->data, submsg_size, byteswap, &sampleinfo, &datap, &datasz))
      goto malformed;
    /* This only decodes the payload when needed (possibly reducing the submsg size). */
    if (!decode_Data (rst->gv, &sampleinfo, datap, datasz, &submsg_len))
      goto malformed;
    /* Set the sample bswap according to the payload info. */
    if (!set_sampleinfo_bswap(&sampleinfo, (struct CDRHeader *)datap))
      goto malformed;
    sampleinfo.timestamp = timestamp;
    sampleinfo.reception_timestamp = tnowWC;
    handle_Data (rst, tnowE, rmsg, &sm->data, submsg_len, &sampleinfo, datap, &deferred_wakeup, prev_smid);
    rst_live = 1;
    ts_for_latmeas = 0;
  }
  break;
```

Checking the Header: The Efficient Way

```
bool MessageReceiver::checkRTPSHeader(
        CDRMessage_t* msg)
{
    //check and proccess the RTPS Header
    if (msg->buffer[0] != 'R' ||  msg->buffer[1] != 'T' ||
            msg->buffer[2] != 'P' ||  msg->buffer[3] != 'S')
    {
        logInfo(RTPS_MSG_IN, IDSTRING "Msg received with no RTPS in header
        return false;
    }

    msg->pos += 4;
```

```
case DATA:
{
    if (dest_guid_prefix_ != participantGuidPrefix)
    {
```

Harness not great: but does the job

```
unsigned char *s = __AFL_FUZZ_TESTCASE_BUF;
size_t sz = (size_t) __AFL_FUZZ_TESTCASE_LEN;
while (__AFL_LOOP(10000))
{
ACE_Message_Block *mb = new ACE_Message_Block (sz);

// do our own write to mb
ACE_OS::memcpy(mb->wr_ptr(), s, sz);
mb->wr_ptr(sz);

// most code below stolen from Spdp.cpp

OpenDDS::DCPS::Serializer ser (mb, encoding_plain_native);
OpenDDS::RTPS::Header header;
if (!(ser >> header)) {
  return 0; // this might be mutated by afl
  // ACE_OS::printf("%s\n", "fail deserialize");
}
```

DoS via malformed RTPS packet (slowloris)

```
    at /usr/local/src/opendds/ACE_wrappers/TAO/tao/Generic_Sequence_T.h:239
#5  0x00007fc3d7dee3e7 in TAO::unbounded_value_sequence<int>::length (this=0x7fff13410178, length=143164
8085) at /usr/local/src/opendds/ACE_wrappers/TAO/tao/Unbounded_Value_Sequence_T.h:62
#6  0x00007fc3d64261c0 in OpenDDS::DCPS::operator>> (strm=..., seq=...) at RtpsCoreTypeSupportImpl.cpp:1
977
#7  0x00007fc3d643630a in OpenDDS::DCPS::operator>> (strm=..., stru=...) at RtpsCoreTypeSupportImpl.cpp:
2335
#8  0x00007fc3d64f7750 in OpenDDS::DCPS::operator>> (outer_strm=..., uni=...) at RtpsCoreTypeSupportImpl
.cpp:11836
#9  0x00007fc3d64e2f18 in OpenDDS::DCPS::operator>> (strm=..., seq=...) at RtpsCoreTypeSupportImpl.cpp:8
985
#10 0x00007fc3d66129bb in OpenDDS::DCPS::operator>> (strm=..., stru=...) at RtpsCoreTypeSupportImpl.cpp:
18452
#11 0x00000000004ce1c7 in main (argc=<optimized out>, argv=<optimized out>) at test.cp
(gdb) l test.cpp:79
74          const size_t start = mb->length();
75          CORBA::UShort submessageLength = 0;
76          switch (subm) {
77            case OpenDDS::RTPS::DATA: {
78              OpenDDS::RTPS::DataSubmessage data;
79              if (!(ser >> data)) {
80                ACE_ERROR((LM_ERROR,
81                      ACE_TEXT("(%P|%t) ERROR: Spdp::SpdpTransport::handle_input() - ")
82                      ACE_TEXT("failed to deserialize DATA header for SPDP\n")));
83                return 0;
```

```
000076c0: 5500 0000 4055 5503 e855 0040 5555 3755  U...@UU..U.@UU7U
000076d0: 5501 ffff ffff e81d 2948 0703 cc00 0003  U.......)H......
000076e0: e055 5562 5555 ffff ffe8 1d29 4807 03cc  .UUbUU.....)H...
000076f0: 5555 0101 0101 0101 0101 0101 0101 1501  UU..............
00007700: 0101 0101 0101 0101 0101 0101 0101 0101  ................
00007710: 0101 0101 0101 0101 0e01 0101 0101 0101  ................
00007720: 0101 0101 0101 0101 0101 0101 0101 0101  ................
00007730: 0101 0101 0101 0101 0101 0101 0101 0101  ................
```

	CVE	CWE	Notes	Status
All	**-**	**CWE-406**	**Network reflection**	
OMG (specs)	**-**	**Extend specs to allow white-listing**		**Disclosed**
RTI ConnextDDS	-	Patched in the most current release		Patched
OCI OpenDDS	CVE-2021-38429	Patched in the most current release		Mitigated with patch
ADLINK CycloneDDS	-	Had already an exp. back-off mechanism		Already mitigated
GurumDDS	-	Had already an exp. back-off mechanism		Already mitigated (No reply, 5 times)
eProsima Fast-DDS	CVE-2021-38429	Patched in the master branch		Mitigated with patch
Twin Oaks CoreDX	N/A	N/A		Disclosed

OCI OpenDDS	**CVE-2021-38445**	**CWE-130**	**Failed assertion check**	**Patched**
OCI OpenDDS	**CVE-2021-38447**	**CWE-405**	**Resource exhaustion (slowloris)**	**Patched**

No source code?
No problem!

Decompile and debug to find good targets

```
Thread 9 "FilterBug" received signal SIGS
[Switching to Thread 0x7fffe67fc700 (LWP
0x00005555559690ca in DDS_DynamicData2Typ
1195    DynamicData2TypePlugin.c: No such
(gdb) bt
#0  0x00005555559690ca in DDS_DynamicData
#1  0x000055555605ab44 in PRESPsReaderQue
#2  0x000055555605f4ac in PRESPsReaderQue
    entry=0x555556da55d0, receptionTsIn=0
    at PsReaderQueue.c:4266
#3  0x0000555556068437 in PRESPsReaderQue
    rejectedReason=0x555556da1198, receiv
    localData=0x0, decodingKeyHandle=0x0,
    worker=0x555556a3a630) at PsReaderQue
#4  0x0000555555cd0f97 in PRESPsService_r
    timestamp=0x7fffe67fb8a0, storage=0x5
#5  0x0000555555e2bf8c in COMMENDBeReaderService_onSubmessage (listener=0x555556a96b30, conte
    at BeReaderService.c:1350
#6  0x0000555555edef4d in MIGInterpreter_parse (me=0x5555568d2830, context=0x7fffcc000be0, ms
#7  0x0000555555e22776 in COMMENDActiveFacadeReceiver_loop (param=0x555556a3a600) at ActiveFa
#8  0x000055555601209c in RTIOsapiThreadChild_onSpawned (param=0x555556716b00) at Thread.c:14
#9  0x00007ffff79b96db in start_thread (arg=0x7fffe67fc700) at pthread_create.c:463
#10 0x00007ffff6f3988f in clone () at ../sysdeps/unix/sysv/linux/x86_64/clone.S:95
```

```
134   }
135   else {
136     if ((*(int *)bufferToParse.pointer == 0x53505452) ||
137         (*(int *)bufferToParse.pointer == 0x58505452)) {
138       data = (char *)((int *)bufferToParse.pointer + 1);
139       iVar11 = MIGInterpreter_parseHeader(context,&data);
140       if (iVar11 == 0) {
141         (me->_stat).versionMismatchCount = (me->_stat).versionMismatchCount + 1;
142       }
143       else {
144         keyToMatch.guid.prefix._0_8_ = *(undefined8 *)&context->sourceHostId;
145         keyToMatch.guid.prefix.instanceId = context->sourceInstanceId;
146         pMVar3 = (me->_property).forwarder;
147         pRVar14 = context->_decodeBuffers;
148         pRVar15 = context->_decodeBuffers + 1;
149         uVar17 = 0x637e93;
```

No source? No problem!

- afl supports QEMU binary instrumentation
- afl-unicorn
  - Unicorn engine instrumentation + afl
  - dump context with gdb, afl-unicorn restores that context, sets reg & memory, afl-unicorn sets RIP and starts execution
  - insert your fuzz payload, start running

afl-unicorn: challenges

- Must locate the correct function
- Re-implement memory operation functions
  - (glibc) malloc, memset, etc
- Speed is not great in Python & how afl-unicorn works

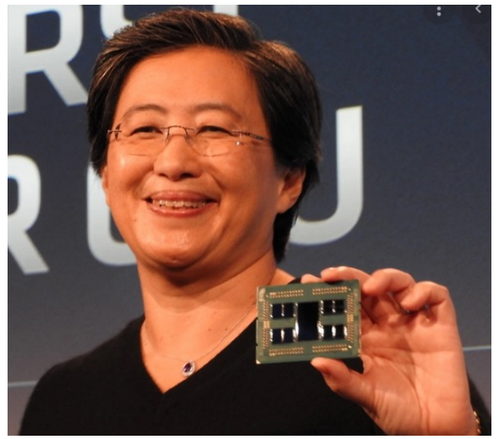
afl-unicorn: speed

- Has to restore context on each iteration

```
                american fuzzy lop 2.52b (master)

process timing ──────────────────────────────┬─ overall results ──
      run time : 3 days, 23 hrs, 33 min, 34 sec │  cycles done : 0
 last new path : 0 days, 7 hrs, 35 min, 58 sec  │  total paths : 2639
             h : 0 days, 5 hrs, 51 min, 17 sec  │ uniq crashes : 556
             g : 0 days, 5 hrs, 18 min, 17 sec  │  uniq hangs : 130
             s ──────────────┬─ map coverage ──
             g : 1562* (59.19%) │    map density : 0.32% / 1.49%
             t : 14 (0.53%)     │ count coverage : 5.06 bits/tuple
             s ──────────────┴─ findings in depth ──
  now trying : interest 32/8    │ favored paths : 288 (10.91%)
stage execs : 15.8k/73.4k (21.47%) │  new edges on : 551 (20.88%)
total execs : 9.48M             │ total crashes : 2.83M (556 unique)
 exec speed : 9.24/sec (zzzz...) │  total tmouts : 61.1k (187 unique)
```

Throw money to the problem

- AWS EC2 m5a.8xlarge (Glorious AMD EPYC, 200%+ faster)
- Affordable by various means (time)
- Alternatively: rewrite harness in C, optimize how context is restored, etc



```
1  [||||||||||100.0%]   9  [||||||||||100.0%]   17 [||||||||||100.0%]   25 [||||||||||100.0%]
2  [||||||||||100.0%]   10 [||||||||||100.0%]   18 [||||||||||100.0%]   26 [||||||||||100.0%]
3  [||||||||||100.0%]   11 [||||||||||100.0%]   19 [||||||||||100.0%]   27 [||||||||||100.0%]
4  [||||||||||100.0%]   12 [||||||||||100.0%]   20 [||||||||||100.0%]   28 [||||||||||100.0%]
5  [||||||||||100.0%]   13 [||||||||||100.0%]   21 [||||||||||100.0%]   29 [||||||||||100.0%]
6  [||||||||||100.0%]   14 [||||||||||100.0%]   22 [||||||||||100.0%]   30 [||||||||||100.0%]
7  [||||||||||100.0%]   15 [||||||||||100.0%]   23 [||||||||||100.0%]   31 [||||||||||100.0%]
8  [||||||||||100.0%]   16 [||||||||||100.0%]   24 [||||||||||100.0%]   32 [||||||||||100.0%]
Mem[||||||||                          2.15G/62.2G]   Tasks: 134, 121 thr; 28 running
Swp[                                     0K/0K]   Load average: 32.45 32.21 32.12
                                                  Uptime: 2 days, 12:14:11
```

	CVE	CWE	Notes	Status
All	**-**	**CWE-406**	**Network reflection**	
OMG (specs)	**-**	**Extend specs to allow white-listing**		**Disclosed**
RTI ConnextDDS	-	Patched in the most current release		Patched
OCI OpenDDS	CVE-2021-38429	Patched in the most current release		Mitigated with patch
ADLINK CycloneDDS	-	Had already an exp. back-off mechanism		Already mitigated
GurumDDS	-	Had already an exp. back-off mechanism		Already mitigated (No reply, 5 times)
eProsima Fast-DDS	CVE-2021-38429	Patched in the master branch		Mitigated with patch
Twin Oaks CoreDX	N/A	N/A		Disclosed

OCI OpenDDS	**CVE-2021-38445**	**CWE-130**	**Failed assertion check**	**Patched**
OCI OpenDDS	**CVE-2021-38447**	**CWE-405**	**Resource exhaustion (slowloris)**	**Patched**
RTI ConnextDDS	**CVE-2021-38435**	**CWE-131**	**Segmentation fault via network**	**Patched**
GurumDDS	**CVE-2021-38423**	**CWE-131**	**Segmentation fault via network**	**No reply (5 times)**
GurumDDS	**CVE-2021-38439**	**CWE-122**	**Heap-overflow via network**	**No reply (5 times)**

black hat EUROPE 2021
november 10-11, 2021
BRIEFINGS

Configuration-based protocol? Let's fuzz the configuration parsers

Federico Maggi Trend Micro Research

Before we begin

- Way to go: **use well-tested XML libraries**
- The OK way: re-implement and test often
- The bad way: use unmaintained XML library

Use unmaintained XML library

Federico Maggi
@phretor

I'm reviewing a software using a core parsing library unmaintained since 2006 (was beta), mailing list silent since 2010. They renamed it and changed the version number from 0.8.5 to 1.0.0 (to make it look more legit?), leaving code *untouched*.

It's fine.

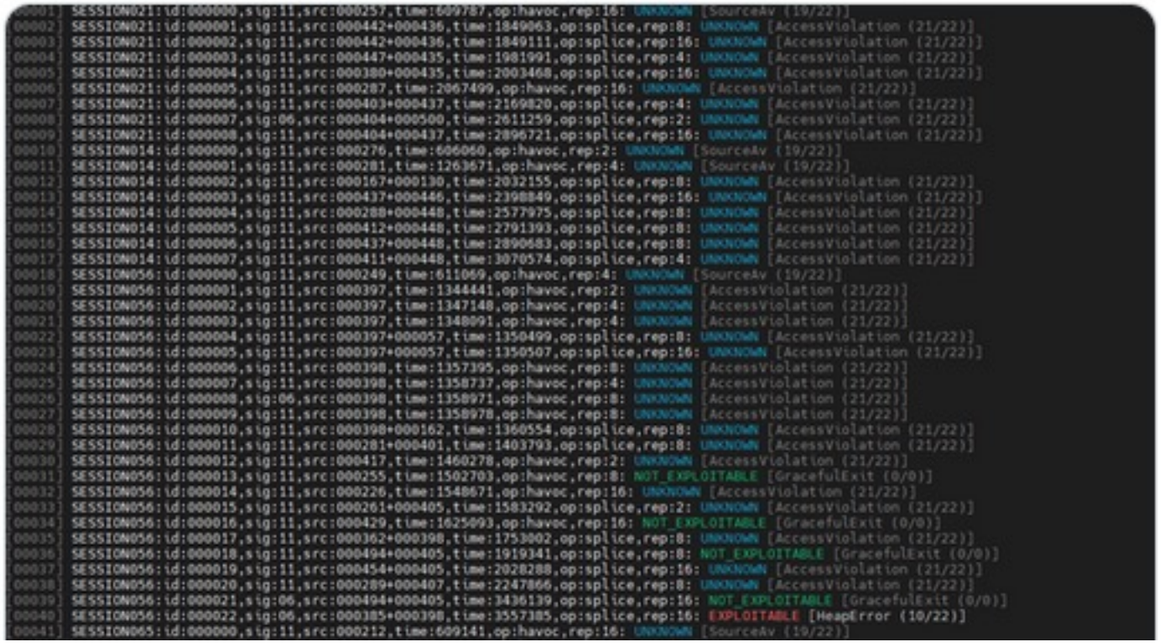
6:08 PM · Jul 22, 2021

12 1 Copy link to Tweet

Tweet your reply

Richard Johnson
@richinseattle

I really would prefer that I didn't find crashes in 90% of the random packages I harness when seeing them mentioned in tweets, but here we are. TP-Link uses ezxml, have some free CVEs, fuzzer friends



Radamsa + scripting: still does the job elegantly

```
@xml

<domain_aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
name="CircleSubParticipant" domain_ref="MyDomainLibrary::MyDomain">
```

```
Program received signal SIGSEGV, Segmentation fault.
0x0000626262626262 in ?? ()
(gdb) bt
#0  0x0000626262626262 in ?? ()
#1  0x00005555559e4aa0 in ?? ()
#2  0x0000555555b69650 in ?? ()
#3  0x0000555555b6cd30 in ?? ()
#4  0x00005555559d8a40 in ?? ()
#5  0x0000555555a2ac60 in ?? ()
#6  0x00007ffff63a912b in RTI_utf8_toUtf8 ()
   from /home/trend/rticonnextdds-connector-py/rticonnextdds_connector/../rticonnextdds-connector/lib/
x64Linux2.6gcc4.4.5/librtiddsconnector.so
#7  0x0000000000000000 in ?? ()
(gdb) i r
rax            0x0      0
rbx            0x6161616161616161       7016996765293437281
rcx            0x64     100
rdx            0x61     97
rsi            0x7fffffffa850   140737488332880
rdi            0x555555a01640   93824997135936
rbp            0x6161616161616161       0x6161616161616161
rsp            0x7fffffffaa90   0x7fffffffaa90
r8             0x6e     110
r9             0x7fffffffb610   140737488336400
r10            0x60     96
r11            0x7ffff7977dc8   140737347288520
r12            0x6161616161616161       7016996765293437281
r13            0x6161616161616161       7016996765293437281
r14            0x6161616161616161       7016996765293437281
r15            0x6161616161616161       7016996765293437281
rip            0x626262626262   0x626262626262
eflags         0x10202  [ IF RF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
(gdb) 
```

Radamsa + scripting: still does the job elegantly

```
@xml

<domain_participant name="MyPubParticipant"
domain_ref="aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

'bbbbbb' part could overwrite RIP

```
Program received signal SIGSEGV, Segmentation fault.
0x0000626262626262 in ?? ()
(gdb) bt
#0  0x0000626262626262 in ?? ()
#1  0x0000000000000000 in ?? ()
(gdb) i r
rax            0x0      0
rbx            0x6161616161616161       7016996765293437281
rcx            0x0      0
rdx            0x61     97
rsi            0x7fffffff9e10   140737488330256
rdi            0x0      0
rbp            0x6161616161616161       0x6161616161616161
rsp            0x7fffffffa190   0x7fffffffa190
r8             0x0      0
r9             0x0      0
r10            0x0      0
r11            0x0      0
r12            0x6161616161616161       7016996765293437281
r13            0x6161616161616161       7016996765293437281
r14            0x6161616161616161       7016996765293437281
r15            0x6161616161616161       7016996765293437281
rip            0x626262626262   0x626262626262
eflags         0x10202  [ IF RF ]
cs             0x33     51
ss             0x2b     43
ds             0x0      0
es             0x0      0
fs             0x0      0
gs             0x0      0
(gdb)
```

Radamsa + scripting: still does the job elegantly

```
@xml

<domain_participant name="CircleSubParticipant"
domain_ref="MyDomaiaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

'aaaaaaaabbbbbbbbcccccccddddddddd' part could overwrite memory address of environment variables

```
Program received signal SIGSEGV, Segmentation fault.
0x00007ffff60991aa in RTIXMLObject_lookUpRef () from /home/trend/rticonnextdds-connector-py/rticonnextdds_connector/../rticonnextdds-connector/lib/x64Linux2.6gcc4.4.5/librtiddsconnector.so
(gdb) x/s *((char **)environ)
0x6161616161616161:     <error: Cannot access memory at address 0x6161616161616161>
(gdb) x/s *((char **)environ+1)
0x6262626262626262:     <error: Cannot access memory at address 0x6262626262626262>
(gdb) x/s *((char **)environ+2)
0x6363636363636363:     <error: Cannot access memory at address 0x6363636363636363>
(gdb) x/s *((char **)environ+3)
0x6464646464646464:     <error: Cannot access memory at address 0x6464646464646464>
(gdb) x/s *((char **)environ+4)
0x7fffffffe000: ""
(gdb) x/s *((char **)environ+5)
0x7fffffffe0dd: "LC_MONETARY=lzh_TW"
(gdb) x/s *((char **)environ+6)
0x7fffffffe0f0: "XDG_GREETER_DATA_DIR=/var/lib/lightdm-data/trend"
(gdb) x/s *((char **)environ+7)
0x7fffffffe121: "SESSION=ubuntu"
(gdb) x/s *((char **)environ+8)
0x7fffffffe130: "GPG_AGENT_INFO=/home/trend/.gnupg/S.gpg-agent:0:1"
(gdb) x/s *((char **)environ+9)
0x7fffffffe162: "SHELL=/bin/bash"
(gdb) x/s *((char **)environ+10)
0x7fffffffe172: "XDG_MENU_PREFIX=gnome-"
(gdb)
```

	CVE	CWE	Notes	Status
All	**-**	**CWE-406**	**Network reflection**	
OMG (specs)	**-**	**Extend specs to allow white-listing**		**Disclosed**
RTI ConnextDDS	-	Patched in the most current release		Patched
OCI OpenDDS	CVE-2021-38429	Patched in the most current release		Mitigated with patch
ADLINK CycloneDDS	-	Had already an exp. back-off mechanism		Already mitigated
GurumDDS	-	Had already an exp. back-off mechanism		Already mitigated (No reply, 5 times)
eProsima Fast-DDS	CVE-2021-38429	Patched in the master branch		Mitigated with patch
Twin Oaks CoreDX	N/A	N/A		Disclosed

OCI OpenDDS	**CVE-2021-38445**	**CWE-130**	**Failed assertion check**	**Patched**
OCI OpenDDS	**CVE-2021-38447**	**CWE-405**	**Resource exhaustion (slowloris)**	**Patched**
RTI ConnextDDS	**CVE-2021-38435**	**CWE-131**	**Segmentation fault via network**	**Patched**
GurumDDS	**CVE-2021-38423**	**CWE-131**	**Segmentation fault via network**	**No reply (5 times)**
GurumDDS	**CVE-2021-38439**	**CWE-122**	**Heap-overflow via network**	**No reply (5 times)**
GurumDDS	**CVE-2021-38437**	**CWE-1104**	**Unmaintained, vulnerable XML lib.**	**No reply (5 times)**
CycloneDDS	**CVE-2021-38441**	**CWE-123**	**Heap-write primitive in XML parser**	**Patched**
CycloneDDS	**CVE-2021-38443**	**CWE-228**	**8-bytes heap-write primitive**	**Patched**
RTI ConnextDDS	**CVE-2021-38427**	**CWE-121**	**Stack-based overflow in XML parser**	**Patched**
RTI ConnextDDS	**CVE-2021-38433**	**CWE-121**	**Stack-based overflow in XML parser**	**Patched**

Attacker and Threat Model

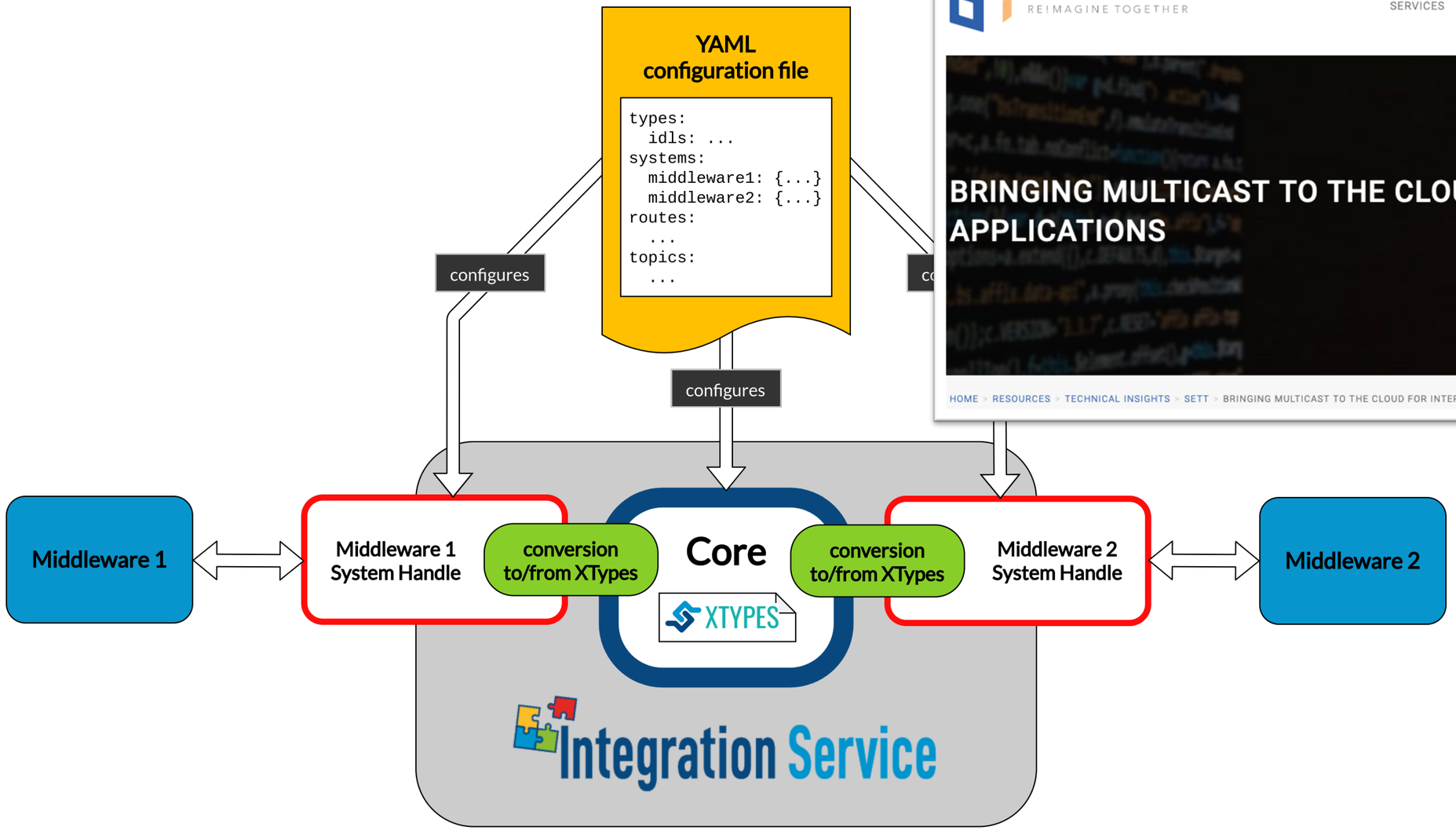
Attacker Model

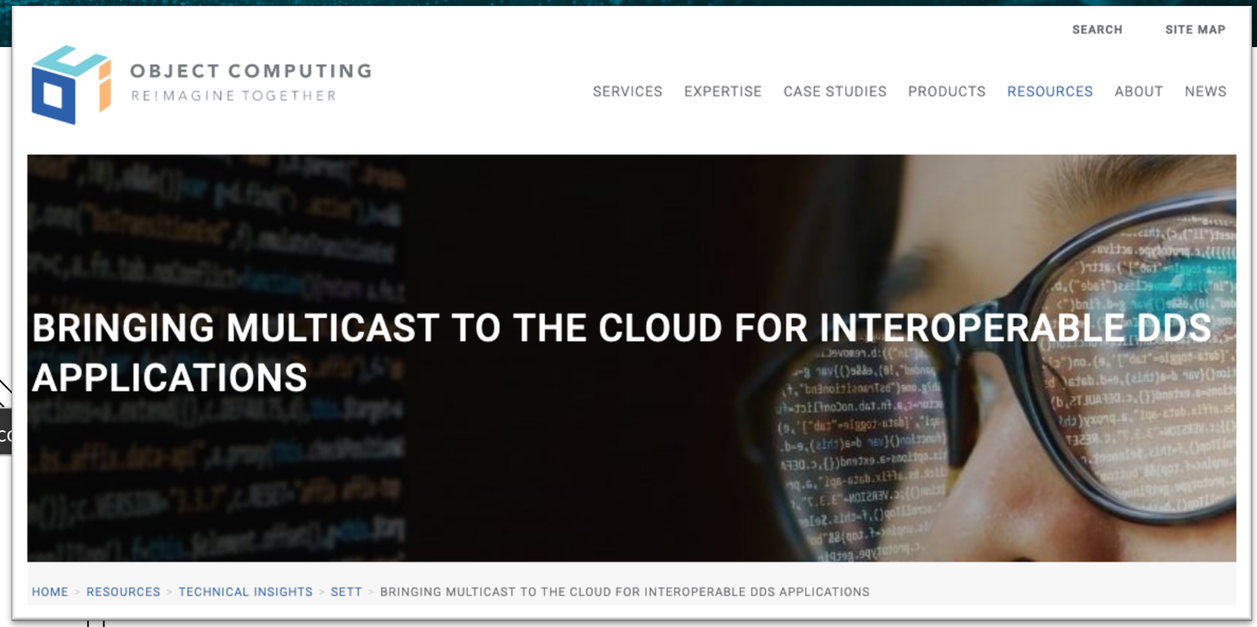
- DDS is deployed **locally** and deep in the control network: don't expect to find exposed endpoints! Exposed ROS2 instances: those, yes.

- DDS creates new opportunities for: **discovery**, **lateral** movment, and **C&C**

- Controlling 1 endpoint is enough to impact other endpoints (see, reflection)

A few words about “remote DDS”





https://integration-service.docs.eprosima.com/en/latest/

Mitigation goes hand to hand with good disclosure when it comes to critical software stacks

Good Mitigation Starts with Good Disclosure

- We **approached vendors in a friendly way** when we **started** working on this project
- Has this "transparency" paid back?
  - **What worked well:** DDS implementors that **helped us write fuzzing harnesses**; how do we push others to do the same?
  - **What didn't work well:** some never responded, even when contacted multiple times from different channels and contact points.
- Should we be treating vendors less like black boxes during the disclosure process?
- **Reflection**: we need more cooperative security assessments.
  - Big round of applause to **ADLINK**, please! 👏👏👏👏👏👏🥳🥳🥳

Future Work & Conclusions

Future Work

- Code base is quite large
  - Implementors, please, reach out and be collaborative: we're here to help!
  - Researchers, please, continue to write new test harnesses!
- Other targets
  - Integration services: we've just scratched the surface
  - DDS Security: we focused only on DDS *core*
  - xTypes

Sound Bytes

- Data Distribution Service (DDS) enables **crucial** technologies
- We discovered and disclosed **vulnerabilities** in most of the OMG-certified DDS implementations
- **Specs** do not prevent **reflection** attacks: implementors should!
- We release
  - an open source layer to parse and forge DDS/RTPS packets in Scapy
  - fuzzing harnesses that we plan to incorporate in the Google OSS-Fuzz initiative right after this talk

black hat EUROPE 2021
november 10-11, 2021
BRIEFINGS

QUESTIONS?

The Data Distribution Service (DDS) Protocol is Critical Let's Use it Securely!

Ta-Lun Yen — **TXOne Networks**
Federico Maggi — **Trend Micro Research**
Erik Boasson — **ADLINK Tech. Labs**

C. Toyama, P. Kuo, and M. Cheng — TXOne Networks
Víctor Mayoral Vilches — Alias Robotics

txOne networks

TREND MICRO | research

ADLINK Leading EDGE COMPUTING

ALIAS ROBOTICS Robot Cybersecurity