



UNIVERSITÀ DI PERUGIA  
Dipartimento di Matematica e Informatica



TESI TRIENNALE IN ...

# Vulnerabilità DDS OMG

*Relatore*

**Prof. Francesco Santini**

*Candidato*

**Federico Ranocchia**

---

Anno accademico 2024/2025

Quì la dedica...

# Indice

<b>1</b>	<b>Introduzione al DDS</b>	<b>5</b>
1.1	Modello publish/subscribe . . . . .	5
1.1.1	Perché non usiamo una connessione con TCP/IP . . . . .	6
1.1.2	Struttura modello publish/subscribe . . . . .	6
1.2	Che cos'è il Data Distribution Service . . . . .	7
1.2.1	Global Space Data . . . . .	8
1.2.2	Quality of Service (QoS) . . . . .	8
1.2.3	Dynamic Discovery . . . . .	8
1.2.4	Modularità ed estensioni . . . . .	8
1.3	Le entità del DDS . . . . .	8
1.3.1	Publisher e Subscriber . . . . .	9
1.3.2	DataWriter e DataReader . . . . .	9
1.3.3	Topic . . . . .	10
1.3.4	Key e Istanza . . . . .	10
1.3.5	Domain . . . . .	11
1.3.6	DomainParticipant . . . . .	11
1.4	Modello DDS . . . . .	12
<b>2</b>	<b>Vulnerabilità DDS</b>	<b>13</b>
2.1	Attacchi DDoS . . . . .	13
2.1.1	DDoS blocco ricezione da parte del datareader Foglio 2 . . . . .	14
2.1.2	DDoS sfruttando estensione DDS security . . . . .	15
2.2	Attacchi di enumerazione e di sniffing . . . . .	16
2.2.1	Enumeration sniff foglio 2 e foglio 5 . . . . .	16
2.3	QoS Exploitation Attack foglio 1 . . . . .	18

2.3.1	Foglio 4-B Modifica maligna di ownership strength . . . . .	19
2.3.2	Foglio 4-D Modifica maligna di LIFESPAN QoS . . . . .	20

# Capitolo 1

## Introduzione al DDS

In questo capitolo viene fatta un'introduzione generale dello standard del Data Distribution Service (DDS) gestita dall'Object Management Group (OMG). Inizialmente a livello generale per poi andare sempre più nel dettaglio per capire il suo funzionamento, che ci sarà utile capire per comprendere le vulnerabilità che verranno analizzate in successivi capitoli. Inoltre verrà introdotta la sua estensione DDS security che si occupa di rendere più sicuro lo standard DDS aggiungendo elementi di sicurezza come l'autenticazione e una implementazione della cifratura dei pacchetti scambiati tra i vari dispositivi connessi alla rete. Successivamente verranno mostrate delle implementazioni e in quali contesti viene utilizzato attualmente e in futuro. Infine verrà mostrato in quali contesti attuali e futuri viene utilizzato con le sue diverse implementazioni.

In questa tesi, in dei casi, verrà omessa la specifica OMG perchè ci riferiremo esclusivamente al DDS conforme all'Object Management Group. Questo standard ha delle specifiche tecniche ben precise, consente l'interoperabilità tra i diversi vendor che lo rispettano, insieme a tanti altri numerosi vantaggi. [3]

### 1.1 Modello publish/subscribe

Prima di parlare del DDS, è necessario prima capire il funzionamento del modello Publish-Subscribe che sta alla base di tutto il suo funzionamento. Questo sistema di publish e subscribe non funziona come la classica applicazione che siamo abituati a vedere tra server e client tramite protocollo TCP/IP nell'ambito delle comunicazioni.

### 1.1.1 Perché non usiamo una connessione con TCP/IP

Prendiamo l'esempio di un collegamento tra una workstation e un sensore per la temperatura. La connessione a livello fisico avverrà tramite un collegamento Ethernet. La workstation e il sensore quindi si trovano nello stesso network e possono ora cominciare a comunicare tra di loro. L'obiettivo è quello di trasferire i dati dal sensore alla workstation in modo tale da poterli visualizzare a schermo. Il metodologia più frequente è quella di utilizzare un socket tramite protocollo TCP/IP, ma non sempre è la soluzione migliore. Dei vantaggi del TCP/IP sono la disponibilità di utilizzo in molte situazioni e nella maggior parte delle connessioni a Internet viene utilizzato questo protocollo. Tuttavia, in certe situazioni, TCP/IP non risulta la soluzione migliore, specialmente quando dobbiamo collegare un numero di dispositivi al network che può cambiare. Se nel nostro network tra workstation e sensore della temperatura, aggiungiamo un'altro dispositivo, come un sensore per la temperatura, bisogna creare un nuovo socket TCP/IP per far comunicare il nuovo sensore con la workstation. Questo succede perché TCP/IP supporta una comunicazione di tipo one-to-one (uno a uno). Come vedremo nella prossima sottosezione, il modello publish/subscribe non ha questa limitazione [11].

### 1.1.2 Struttura modello publish/subscribe

Per funzionare il modello publish/subscribe ha bisogno di elementi che sono necessari per il sistema di scambio delle comunicazioni. Questi elementi, che rappresentano i componenti all'interno del modello publish/subscribe, vengono chiamati chiamati entità. Il publisher e il Subscriber sono le entità principali che possono comunicare tra di loro quando hanno un topic, che rappresenta una tipologia di dati, (ad esempio temperatura, distanza, velocità, etc...) in comune tra di loro.

- Publisher: colui che "pubblica" nuovi dati riguardanti dei topic rendendoli accessibili ai subscriber iscritti. Di solito si tratta di un sensore.
- Subscriber: colui che si "iscrive" ai topic del publisher, cominciando così a ricevere nuovi dati sul topic scelto. Molte volte si tratta di un dispositivo utilizzato per mostrare informazioni, come un semplice schermo.

Una caratteristica di queste entità é che possono essere aggiunte o rimosse senza nessun problema, sia publisher che subscriber, dato che le comunicazioni avvengono in modalità asincrona. Un publisher come impostazione predefinita non deve ricevere conferma di ricezione da parte del subscriber a cui manda i pacchetti contenenti i dati riguardo il topic. In questo modo il publisher può continuamente mandare nuovi dati senza effettuare operazioni di conferma, rendendo così le comunicazioni molto più responsive [3].

Un'altra qualità del modello publish/subscribe si manifesta quando un publisher deve inviare dei nuovi dati a più subscriber. In tale situazione, un modello multicast viene utilizzato per i mandare i pacchetti, che vengono poi ricevuti da un'entità che si occuperà di mandare ai subscriber le nuove informazioni riguardanti i topic a cui sono iscritti. Il modello così risulta molto flessibile e utilizzabile in ambienti real-time dove le fonti delle comunicazioni possono cambiare o essere utilizzate da più dispositivi, ad esempio i subscriber possono avere cambiare i topic a cui sono iscritti e/o i publisher possono smettere di pubblicare nuove informazioni su un determinato topic [5].

## 1.2 Che cos'è il Data Distribution Service

Il DDS gestito da OMG è un middleware e uno standard API per una gestione dei dati di tipo data-centric. Questo middleware è un software che si trova tra l'applicativo e il sistema operativo. Viene utilizzato un modello di tipo publish/subscribe, ma con un approccio più data-centric, in modo tale da semplificare il lavoro del programmatore che si deve solamente occupare di specificare il contesto del dato che deve mandare o ricevere. Così facendo non bisogna preoccuparsi dell'invio o della ricezione dei messaggi, perchè questa parte viene completamente gestita dal middleware. Prima di questo middleware esistevano solamente soluzioni proprietarie che si basavano sempre sul modello publish/subscribe. Il DDS é stato il primo standard a formalizzare le comunicazioni di tipo data-centric. Inizialmente erano disponibili solamente soluzioni proprietarie senza specificare un standard univoco. Non avendo avuto uno standard univoco le varie implementazioni non sono compatibili tra di loro a differenza del DDS che lo é.

Altri vantaggi del DDS includono una architettura adattabile dato che supporta degli elementi di auto-scoperta (auto-discovery) in modo tale da aggiungere o rimuov-

vere dispositivi dalla rete in automatico. Rispetto al protocollo TCP/IP il DDS ha la possibilità di effettuare comunicazioni di tipo uno a molti (one-to-many), molti a molti (many-to-many) e uno a uno (one-to-one) [11].

### **1.2.1 Global Space Data**

Il DDS a livello concettuale utilizza un concetto chiamato Global Data Space (spazio dati globale) che consente agli applicativi di accedere a una sorta di memoria locale tramite API. L'applicativo nella scrittura o nella ricezione dei dati utilizzerà questa memoria locale fittizia come un'unica risorsa centralizzata. In realtà, i dati all'interno di questa memoria possono contenere informazioni provenienti da nodi remoti distribuiti per la rete. L'applicativo non deve così preoccupandosi dell'accessibilità dei dati, dato che vengono gestiti dal DDS.[7]

### **1.2.2 Quality of Service (QoS)**

Per soddisfare i diversi requisiti di una trasmissione dati, il Data Distribution Service (DDS) utilizza un insieme di policy di Quality of Service (QoS). Queste policy permettono di controllare, regolare e ottimizzare lo scambio di dati tra i vari componenti all'interno del middleware. Le policy QoS possono variare significativamente in base al tipo di comunicazione richiesta, offrendo una gestione altamente flessibile e granulare. Ogni elemento del middleware può essere configurato con policy specifiche, consentendo un adattamento preciso alle esigenze dell'applicazione.

### **1.2.3 Dynamic Discovery**

### **1.2.4 Modularità ed estensioni**

Ad esempio DDS security

## **1.3 Le entità del DDS**

Il DDS per operare utilizza le entità, che rappresentano gli elementi necessari per il funzionamento del middleware. Hanno il compito di gestire i dati scambiati tra i vari partecipanti all'interno del sistema. Le entità principali del DDS sono: il publisher, il



subscriber, il `DataWriter`, il `DataReader`, il `Topic`, la `Istanza`, il `Domain` e il `Domain Participant`. Ognuna di queste identità deve tener conto del set suo di policies QoS configurate che ne definiscono il comportamento. Queste policy verranno analizzate nel più nel dettaglio in una prossima sezione.

Il linguaggio utilizzato da queste entità si chiama `Interface Definition Language (IDL)`, viene gestito anch'esso da `OMG`. Molto simile a un linguaggio tipizzato simile a `C++`, supporta data types come `char`, `int`, `double`, `float` etc [11] ...

### 1.3.1 Publisher e Subscriber

Qualche entità l'abbiamo già incontrata in precedenza nel modello `publish/subscribe`, in particolare, il `publisher` e il `subscriber` che mantengono immutato il loro ruolo all'interno del `DDS`.

### 1.3.2 DataWriter e DataReader

Per comunicare tra loro, il `publisher` si interfaccia tramite `DataWriter`, mentre il `subscriber` si interfaccia tramite `DataReader`. Il `DataWriter` e il `DataReader` si definiscono come interfacce, poiché l'applicativo può mandare e/o ricevere dati all'interno del middleware tramite queste due entità fondamentali. Questi dati scambiati tra middleware e applicativo durante la spedizione e/o ricezione sono i `data type` e i `data-object`. I `data type` consentono di descrivere la struttura e il formato del dato, mentre i `data-object` sono i dati veri e propri dato che rispettano le specifiche `data type`.

- `DataWriter`: è l'interfaccia usata dagli applicativi per comunicare i `data-objects` con un loro specifico `data type` ai `publisher`. Ricevuti questi `data-objects` il `publisher` potrà mandare le informazioni ricevute dall'applicativo ai relativi `subscriber`.
- `DataReader`: è l'interfaccia usata dagli applicativi per ricevere i `data-objects` con i loro `data type` pubblicati in precedenza da un `publisher`.

[3] Inoltre c'è la possibilità di avere più `DataWriter` associati a un `publisher` e più `DataReader` associati a un `Subscriber`. Tuttavia, un `DataWriter` o un `DataReader` possono essere associati solamente ad un solo `topic`. Ad esempio avendo i `topic`:

velocità e temperatura, dobbiamo utilizzare due DataWriter e due DataReader per effettuare una comunicazione tra publisher e subscriber dato che possono gestire un solo data type per volta.

### 1.3.3 Topic

Nel modello publish/subscribe abbiamo già introdotto i topic, ma abbiamo la necessità di approfondirli quando vengono utilizzati con le specifiche del DDS.

I topic vengono utilizzati per identificare il tipo di dato che viene scambiato tra i publishers e i subscribers, creando così un punto di connessione tra DataWriter e DataReader[6]. All'interno troviamo il nome del topic (che deve essere nel dominio dove si trova), i data types del topic e le policy QoS relative ai topic. Il nome del topic deve essere una stringa che serve a identificare il topic all'interno di un dominio e quindi deve essere univoca.

### 1.3.4 Key e Istanza

Uno o più data types di un topic possono diventare una chiave (key) per quel particolare topic. Queste chiavi ci permettono di suddividere meglio i dati per un singolo topic, dividendo i dati che hanno il valore contenuto nella key diverso. Ogni suddivisione che effettuiamo tramite una key crea un'istanza diversa. L'istanza al suo interno contiene i data-values che vengono ricevuti da un qualche flusso di data (data-stream). [8].

Per fare un esempio prendiamo il topic velocità in un contesto dove si vogliono analizzare i dati di una gara. La struttura del topic avrà due elementi: il primo corrisponde al valore della velocità registrata, mentre il secondo mostra l'id-macchina per capire da quale vettura i dati provengono.

Codice 1.1: Esempio di Topic con una key usando il linguaggio IDL

```
struct Veicolo { // Nome del topic
    id_macchina; // Key del topic
    velocita;
}
```

Creiamo ora due istanze una con valore 270 (velocità) e con l'id-macchina a 1 e l'altra con un valore di 220 e con id-macchina uguale a 2. Notiamo che l'id-macchina in questo caso fungerà da key dell'istanza per distinguere la provenienza dei dati, in questo modo possiamo controllare le due macchine con due istanze ciascuna.

### 1.3.5 Domain

L'entità del dominio rappresenta uno spazio logico definito con lo scopo di contenere Topics e data-types. Le proprietà del domain sono:

- Ogni dominio viene identificato da un id per renderlo univoco.
- Ogni entità del DDS può appartenere a un solo dominio
- Le entità all'interno del dominio possono interagire solamente con le altre entità all'interno dello stesso domain
- Due applicativi DDS per poter comunicare tra di loro hanno bisogno di entrare in un unico dominio
- Un applicativo, creando l'entità DomainParticipant per ogni dominio a cui vuole partecipare, può far parte di più di un dominio

[10]

### 1.3.6 DomainParticipant

L'entità del DomainParticipant viene utilizzata da un'applicazione per entrare all'interno di un domain. Rappresenta la prima entità creata da un'applicazione che verrà utilizzata per creare altre entità. Ha il compito di inizializzare le comunicazioni con il domain attraverso il processo di discovery. Il processo di discovery consente alle entità appartenenti allo stesso domain di trovarsi e connettere automaticamente. Le proprietà del DomainParticipant sono:

- Un DomainParticipant può entrare solamente all'interno di un domain.
- Il DomainParticipant è responsabile per scoprire altre entità DomainParticipant all'interno del domain (dove entrambe si devono trovare)

[9]

## 1.4 Modello DDS

[illegible]

# Capitolo 2

## Vulnerabilità DDS

In questo capitolo ci occuperemo di analizzare e comprendere delle vulnerabilità del protocollo DDS standard OMG (Object Management Group). In particolare verrà analizzato il vettore d'attacco, il protocollo utilizzato, il bersaglio dell'attacco e infine verrà proposta una soluzione applicabile per mitigare possibili attacchi non autorizzati. Nel prossimo capitolo grazie all'aiuto del software –inserire software– riusciremo a capire come queste vulnerabilità possono essere ricreate in un ambiente simulato. Queste vulnerabilità hanno una base di appoggio solida per l'attaccante. In molti casi un dispositivo ha già la possibilità di controllo di un partecipante all'interno della rete o ha la possibilità di modificare dei file di configurazione all'interno del network stesso.

Queste vulnerabilità riguardano la versione del DDS 1.4 con le specifiche dello standard OMG.

### 2.1 Attacchi DDoS

Questi attacchi consistono nel sovraccaricare uno o più dispositivi collegati alla rete DDS in modo tale da renderli non responsivi. Infatti molti sono di tipo I O T – e la potenza di calcolo nella maggior parte dei casi è ridotta. Per di più in molti casi vengono utilizzati dispositivi che non possono permettersi –delay– nell'analisi di certi dati, specialmente in ambiti RealTime in cui bisogna avere delle risposte rapide, come ad esempio nel campo della medicina e nel campo militare.

### 2.1.1 DDoS blocco ricezione da parte del datareader Foglio 2

Citazioni da foglio 2 a gogo Il vettore di attacco si trova nel protocollo RTPS che si occupa di scambiare pacchetti tra i DataReader (coloro che si iscrivono ai vari ai vari topic) e i DataWriter (di solito sono sensori che inviano dati). Questo protocollo utilizza il messaggio HEARTBEAT che viene mandato da un DataWriter a un DataReader per specificare il sequence number del DataWriter. Il sequence number serve al DataReader per sincronizzarsi con il DataWriter durante la ricezione dei pacchetto. Infatti il DataReader quando riceve il sequence number all'interno di un HEARTBEAT può identificare se ci sono o no dei pacchetti mancanti e in caso segnalarli al DataWriter.[13]

Un DataWriter inoltre può richiedere un messaggio ACKNACK da un DataReader se nell'HEARTBEAT inviato in precedenza dal DataWriter il parametro FINAL è attivo. Il messaggio ACKNACK consente di far rimanere sempre sincronizzato il DataReader al DataWriter che non potrà spedire nuovi pacchetti HEARTBEAT fino a quando non avrà ricevuto la conferma di ricezione con un messaggio ACKNACK. I controlli del sequence number all'interno dell'HEARTBEAT non sono sufficienti per coprire la rete da questo tipo di attacco:

- un primo controllo viene effettuato per verificare che non ci siano valori negativi
- un altro controllo serve a determinare se l'ultimo sequence number appena ricevuto non ha un valore più alto di quello ricevuto in precedenza

[13]

#### Dettagli attacco AFTER

Per sfruttare questa vulnerabilità l'attaccante deve utilizzare qualche strumento per sniffare la comunicazione tra il DataReader e il DataWriter, intercettando i messaggi HEARTBEAT. Dopo aver catturato un pacchetto di tipo HEARTBEAT e modificato il suo sequence number assegnandogli un valore molto alto, l'attaccante lo invia al DataReader. Una volta ricevuto il pacchetto il DataReader si metterà in attesa di un HEARTBEAT con un sequence number superiore a quello appena ricevuto. Di conseguenza il DataReader non elaborerà più i messaggi legittimi mandati dal DataWriter,

dato che hanno sequence number più piccoli, bloccando così la sua esecuzione indefinitamente. Solo un messaggio HEARTBEAT con un sequence number maggiore a quello del DataReader farà ripristinare la sua esecuzione.[13]

## **Conclusioni AFTER**

Di solito questo tipo di attacco è difficile da identificare. Un messaggio HEARTBEAT riguarda un solo topic, quindi il resto delle comunicazioni che avvengono su topic differenti o anche sullo stesso topic, ma con un DataReader diverso, non subiranno cambiamenti.

### **2.1.2 DDoS sfruttando estensione DDS security**

Per l'attacco sopra citato dobbiamo considerare il modulo del DDS chiamato DDS security versione 1.1.(fonti ora da foglio 6) Questo si occupa di stabilire una connessione sicura tra i vari dispositivi della rete, infatti verranno utilizzati dei plugin da parte dei partecipanti che servono a:

- autenticazione
- controllo accesso
- crittografia
- login
- data tagging

[4] (foglio 3 pag 718)Per effettuare l'autenticazione un partecipante deve risolvere una challenge crittografica richiesta dal sistema di autenticazione della rete. Effettuato poi questo calcolo crittografico, il risultato viene controllato dal sistema di autenticazione per verificare se esso corrisponda all'hash del risultato della challenge crittografica.[12]

Questo attacco è stato scoperto con Proverif, un tool che viene usato per individuare vulnerabilità nei protocolli crittografici. È stato utilizzato in molti studi, come ad esempio nell'analisi della posta elettronica certificata e nell'analisi del TLS 1.3.[1]

### **Dettagli attacco AFTER foglio 3**

L'attacco DDoS avviene durante la fase di autenticazione del protocollo DDS security 1.1, in particolare quando un nuovo dispositivo tenta di collegarsi alla rete e manda una richiesta di autenticazione all'ente di controllo. La richiesta del partecipante viene poi intercettata dall'attaccante che modifica i valori della challenge crittografica all'interno del pacchetto. Modificando ripetutamente questi valori, l'attaccante inizia a inviare molteplici richieste crittografiche alla sua vittima. Il partecipante comincerà a calcolare queste challenge per effettuare l'autenticazione, consumando tutte le sue risorse. Dato che, la vittima è probabilmente un dispositivo IoT che non dispone di una potenza di calcolo molto elevata, si ritroverà occupata per tutto il tempo necessario a risolvere le challenge crittografiche ricevute dall'attaccante, bloccando così il suo funzionamento.[12]

### **Conclusioni AFTER foglio 3**

Una raccomandazione per mitigare questo attacco può essere quello di cambiare delle policy QoS impostando un tempo limite massimo per effettuare l'autenticazione. Queste policy possono fare in modo che i partecipanti non si ritrovino sopraffatti dalle troppe richieste di autenticazione. Un allarme potrebbe essere anche utile per identificare possibili tentativi DDoS di questo tipo, allertando così un amministratore. [12]

## **2.2 Attacchi di enumerazione e di sniffing**

Dal foglio 2 Prendere informazioni DDS senza effettuare veri e propri attacchi di tipo attivo può essere molto utile per un attaccante che prova a penetrare una rete DDS. In molti casi tutto quello che deve fare l'attaccante è osservare i messaggi che vengono scambiati all'interno del network. Successivamente quando si ottengono informazioni a sufficienza sarà più facile per l'attaccante trovare un vettore di attacco.[13]

### **2.2.1 Enumeration sniff foglio 2 e foglio 5**

Prendendo in considerazione, il protocollo RTPS e il suo modulo discovery, possiamo notare che di default sono molto "verbose", cioè scambiano informazioni in chiaro



durante le comunicazioni tra i vari dispositivi.[13] Il modulo discovery del protocollo RTPS a sua volta si suddivide in altri 2 protocolli fondamentali, che sono necessari per le specifiche DDS:

- Simple Participant Discovery Protocol (SPDP)
- Simple Endpoint Discovery Protocol (SEDP)

Per questo attacco ci focalizzeremo in particolare su SPDP che serve ad individuare la presenza dei partecipanti alla rete. In particolar modo il funzionamento si basa su un messaggi di tipo multicast e unicast che vengono mandati a tutti i dispositivi del network per informare chi è presente attualmente. [3]

### **Dettagli attacco AFTER**

Utilizzando un software in grado di "sniffare" i vari pacchetti della rete, come un semplice script python è stato possibile analizzare il loro contenuto. I pacchetti analizzati sono quelli di tipo multicast RTPS-SPDP. All'interno di un pacchetto di questo tipo possiamo trovare: (nel foglio 2 non viene specificato bene di quale pacchetto si parla, ma guardando la documentazione da pag 125 del foglio 5, stiamo analizzando il pacchetto SPDPdiscoveredParticipandData) (da scrivere in corsivo) l'indirizzo ip dell'host, il prefisso GUID dell'RTPS, la versione dell RTPS, L'ID del venditore, informazioni riguardanti la sincronizzazione ed infine il contenuto dei submessages.[13]

### **Conclusioni AFTER**

Di solito questo tipo di attacco è difficile da identificare e possono essere effettuati anche non avendo un dispositivo autenticato all'interno della rete.

Una soluzione potrebbe essere usare l'estensione del protocollo DDS security o eseguire la connessione tra i nodi tramite un tunnel con WireGuard per criptare le comunicazioni.

## 2.3 QoS Exploitation Attack foglio 1

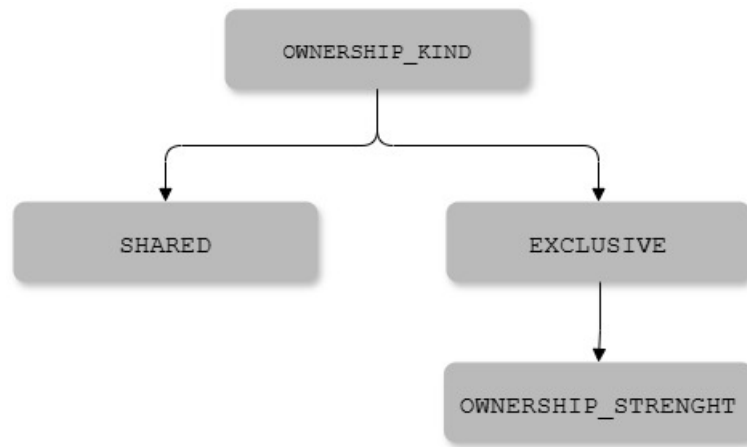


Figura 2.1: Illustrazione policy QoS del DDS

Queste tipologie di attacco sono possibili solo se certe policy QoS vengono modificate durante l'esecuzione della rete, specialmente il parametro OWNERSHIP-KIND che gestisce quanti DataWriter possono scrivere per un determinato Topic. Questo parametro può essere impostato in due modi diversi:

- **SHARED**: in questo modo più di un DataWriter possono aggiornare le informazioni di un topic. Inoltre un DataReader si può iscrivere a qualsiasi scrittore dello stesso topic.
- **EXCLUSIVE**: solo un DataWriter può aggiornare le informazioni di un topic. Il DataWriter che ha il permesso di scrittura per il topic è quello che dispone di un OWNERSHIP-strength con valore più alto.

Un'altra policy QoS che può essere usata come vettore di attacco è quella che regola il parametro LIFESPAN. Questa corrisponde al tempo limite massimo per la lettura da parte di un DataReader di un dato di un topic, che viene inserito all'interno del pacchetto inviato dal DataWriter. Per determinare se un pacchetto di un determinato topic è scaduto viene utilizzato il timestamp di creazione aggiungendo il LIFESPAN impostato; se questo "expiration time" risulta superiore all'orario durante la ricezione del DataReader allora l'informazione ricevuta è ancora valida. Per

funzionare gli orologi del DataWriter e del DataReader devono essere sincronizzati tra di loro.

Un'altra importante policy da considerare è quella riguardo all'affidabilità (RELIABILITY) dei dati riguardanti un topic che può essere impostata in due modi:

- **RELIABLE**: questa impostazione costringe il DataReader a farsi ritrasmettere dal DataWriter i pacchetti mancanti o ricevuti in maniera errata. In questo modo le informazioni del DataReader saranno sempre corrette anche se non sempre saranno aggiornate in tempo reale.
- **BEST-EFFORT**: l'impostazione predefinita non consente il recupero dei pacchetti mancanti o corrotti del DataReader, quindi, quest'ultimo potrebbe anche perdere dei pacchetti che gli sono stati inviati.

[3]

### **2.3.1 Foglio 4-B Modifica maligna di ownership strength**

In una rete dove si utilizza un OWNERSHIP-kind di tipo EXCLUSIVE è possibile utilizzare l'OWNERSHIP-strength a favore dell'attaccante. Infatti è possibile far ricevere informazioni a un DataWriter in maniera errata, dato che quest'ultimo non riceverà più informazioni da una fonte affidabile.[2]

#### **Foglio 4-B Dettagli attacco**

L'attaccante, con un DataWriter in suo possesso all'interno di una rete DDS, può sfruttare il fatto che il topic preso di mira può essere aggiornato solo dal DataWriter con l'OWNERSHIP-strength più alta. Per effettuare questo attacco, tutto quello che serve, è sapere il topic che si vuole modificare, le policy QoS in uso e il valore dell'ownership-strength. L'ultimo passo è quello di impostare il topic scelto nel DataWriter dell'attaccante con OWNERSHIP-strength superiore a quello utilizzato dal DataWriter originario. Ora i DataReader che sono iscritti al topic bersaglio ricevono le informazioni dal DataWriter dell'attaccante. [2]

## Foglio 4-B Conclusioni

L'OWNERSHIP-kind di tipo EXCLUSIVE è utilizzata in contesti dove le informazioni ricevute dal DataReader devono essere accurate dato che un singolo scrittore (in molti casi si tratta di un sensore) può mandare nuovi aggiornamenti del topic. Se l'attaccante, dovesse riuscire a modificare i valori del topic con questo attacco, potrebbe causare in certi casi molti danni, specialmente se il DataWriter dell'attaccante riuscisse a mandare degli aggiornamenti del topic senza essere scoperto. [2]

Una soluzione utile a risolvere questo vettore di attacco potrebbe essere l'utilizzo dell'estensione DDS security che rende impossibile capire qual è il topic bersaglio perchè i messaggi scambiati tra DataReader e DataWriter sono criptati.

### 2.3.2 Foglio 4-D Modifica maligna di LIFESPAN QoS

L'attaccante a volte potrebbe modificare le policy QoS riguardanti il LIFESPAN e se necessario il parametro RELIABLE. Infatti il tempo limite di scadenza dei pacchetti, contenenti i dati del topic, può essere impostato a valori molto piccoli creando problemi di comunicazione tra un DataWriter e un DataReader. Utilizzando un'affidabilità di tipo RELIABLE si riesce a mitigare l'attaccante che così deve utilizzare valori più estremi per compromettere la comunicazione. Questo test è stato dimostrato con RTI Shapes Demo che implementa una soluzione DDS di RTI corrispondente alle specifiche dello standard OMG. [2]

#### Foglio 4-D Dettagli attacco

Avendo sotto controllo questi due parametri policy, l'attaccante può modificare la policy dei DataWriter in modo tale da avere un LIFESPAN molto piccolo. Così facendo, i pacchetti spediti dal publisher arriveranno già scaduti e non potranno più essere utilizzati dai DataReader. In certi casi il pacchetto che deve essere inviato viene distrutto dallo stesso DataWriter all'interno della sua coda prima dell'invio. In questo caso il test è stato effettuato impostando il valore di LIFESPAN  $< 80\text{ms}$  dove si è visto che nessun pacchetto raggiunge il DataReader. Se si aumenta il valore tra gli 80ms e i 100ms già si può notare che dei pacchetti vengono letti con successo dal DataReader, mentre altri vengono eliminati prima della lettura. Infine impostando

un valore LIFESPAN  $\geq 120\text{ms}$  si può notare che la comunicazione tra publisher e subscriber avviene senza nessun problema.

Un dettaglio da aggiungere è che se su RTI Shapes veniva impostata la policy dell'affidabilità (RELIABILITY) di tipo RELIABLE i millisecondi utilizzati dal LIFESPAN per compromettere le comunicazioni tra DataReader e DataWriter devono essere moltiplicati per un fattore di 0.01. Quindi, ad esempio se si ottiene un completo annullamento delle comunicazioni con un LIFESPAN  $< 80\text{ms}$  utilizzando la RELIABILITY di tipo BEST-EFFORT, per ottenere lo stesso risultato con RELIABILITY di tipo RELIABLE dobbiamo impostare un LIFESPAN  $< 0.8\text{ms}$ . [2]

#### Foglio 4-D Conclusioni

Inizialmente molte reti DDS hanno impostato la RELIABILITY di tipo BEST-EFFORT che è l'impostazione predefinita. Quindi nella maggior parte dei casi l'attaccante non si deve preoccupare di questo parametro.

Una possibile soluzione sarebbe quella di impostare qualche tipo di controllo in modo tale da avvertire un operatore umano se molti pacchetti vengono scartati perché arrivati con un LIFESPAN scaduto. Questo controllo potrebbe essere anche utile, nel caso in cui il DataWriter e il DataReader si trovassero distanti fisicamente tra di loro, per verificare la qualità del collegamento. [2]

Tipo di attacco	Vettore attacco	Protoc./ Estens.	Bersaglio nella rete	Software	Soluzione
Discovery devices[13]	Verbose nature of RTPS	DDSI-RTPS	Tutti i partecipanti	Sniffer python	WireGuard
DDos[13]	Heartbeat sequence number	DDSI-RTPS	DataReader	Sniffer python	-
DDoS[12]	Authentication challenge	DDS security 1.1 Discovery protoc.	Tutti i partecipanti	Proverif	Scadenza richieste di autenticazione
QoS policy[2]	ownership-strength	DDSI-RTPS	DataReader	RTI shapes	DDS security
QoS policy[2]	LIFESPAN	DDSI-RTPS	DataReader	RTI shapes	Controllo per LIFESPAN scartati

Tabella 2.1: La versione DDS in tutti i casi è la 1.4

# Elenco delle figure

2.1	Illustrazione policy QoS del DDS . . . . .	18
-----	--	----

# Elenco delle tabelle

2.1	La versione DDS in tutti i casi è la 1.4 . . . . .	21
-----	--	----

# Bibliografia

- [1] Bruno Blanchet, Ben Smyth, Vincent Cheval, and Marc Sylvestre. *ProVerif 2.05: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial*, 2023. URL <https://bblanche.gitlabpages.inria.fr/proverif/manual.pdf>. [Accesso: 2 febbraio 2025].
- [2] Michael James Michaud, Thomas R. Dean, and Sylvain P. Leblanc. Attacking OMG data distribution service (DDS) based real-time mission critical distributed systems. In *13th International Conference on Malicious and Unwanted Software, MALWARE 2018, Nantucket, MA, USA, October 22-24, 2018*, pages 68–77. IEEE, 2018. doi: 10.1109/MALWARE.2018.8659368. URL <https://doi.org/10.1109/MALWARE.2018.8659368>.
- [3] Object Management Group. OMG Data Distribution Service, April 2015. URL <http://www.omg.org/spec/DDS/1.4/PDF>. [Accesso: 2 febbraio 2025].
- [4] Object Management Group. DDS Security, July 2018. URL <https://www.omg.org/spec/DDS-SECURITY/1.1/PDF>. [Accesso: 2 febbraio 2025].
- [5] Sangyoon Oh, Jai-Hoon Kim, and Geoffrey Fox. Real-time performance analysis for publish/subscribe systems. *Future Generation Computer Systems*, 26(3):318–323, 2010. ISSN 0167-739X. doi: <https://doi.org/10.1016/j.future.2009.09.001>. URL <https://www.sciencedirect.com/science/article/pii/S0167739X09001344>.
- [6] OMG. Topic [DDS Foundation Wiki], February 2025. URL [https://www.omgwiki.org/ddsf/doku.php?id=ddsf:public:guidebook:06\\_append:glossary:t:topic](https://www.omgwiki.org/ddsf/doku.php?id=ddsf:public:guidebook:06_append:glossary:t:topic). [Online; accessed 5. Feb. 2025].



- [7] OMG. What is DDS?, February 2025. URL <https://www.dds-foundation.org/what-is-dds-3>. [Online; accessed 6. Feb. 2025].
- [8] RTI. Instance | data distribution service (dds) community rti connext users, February 2025. URL <https://community.rti.com/glossary/instance>. [Online; accessed 2025-02-05].
- [9] RTI. DomainParticipant | Data Distribution Service (DDS) Community RTI Connex Users, February 2025. URL <https://community.rti.com/glossary-term/domainparticipant>. [Online; accessed 6. Feb. 2025].
- [10] RTI. Domain | Data Distribution Service (DDS) Community RTI Connex Users, February 2025. URL <https://community.rti.com/glossary/domain>. [Online; accessed 6. Feb. 2025].
- [11] J.M. Schlesselman, G. Pardo-Castellote, and B. Farabaugh. Omg data-distribution service (dds): architectural update. In *IEEE MILCOM 2004. Military Communications Conference, 2004.*, volume 2, pages 961–967 Vol. 2, 2004. doi: 10.1109/MILCOM.2004.1494965.
- [12] Bingham Wang, Hui Li, and Jingjing Guan. A formal analysis of data distribution service security. In Jianying Zhou, Tony Q. S. Quek, Debin Gao, and Alvaro A. Cárdenas, editors, *Proceedings of the 19th ACM Asia Conference on Computer and Communications Security, ASIA CCS 2024, Singapore, July 1-5, 2024*. ACM, 2024. doi: 10.1145/3634737.3656288. URL <https://doi.org/10.1145/3634737.3656288>.
- [13] Thomas White, Michael N. Johnstone, and Matthew Peacock. An investigation into some security issues in the dds messaging protocol, 2017. URL <https://api.semanticscholar.org/CorpusID:52840449>.

# Ringraziamenti

Lorem ipsum dolor sit amet, consectetur adipisci elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrum exercitationem ullamco laboriosam, nisi ut aliquid ex ea commodi consequatur. Duis aute irure reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. Excepteur sint obcaecat cupiditat non proident, sunt in culpa qui officia deserunt mollit anim id est laborum

Caio