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

1 Introduzione al DDS **5**

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'é il Data Distribution Service 7

1.2.1 Quality of Service (QoS) 7

1.2.2 Global Space Data 8

1.2.3 Architettura DDS 8

1.3 Le entitá del DCPS 9

1.3.1 Publisher e Subscriber 9

1.3.2 DataWriter e DataReader 9

1.3.3 Topic 10

1.3.4 Key e Instanza 10

1.3.5 Domain 11

1.3.6 DomainParticipant 11

1.4 Le policy QoS nel dettaglio 12

1.5 Il protocollo RTPS 13

1.5.1 Structure module 13

1.5.2 Behavior module 13

1.5.3 Messages module 14

1.5.4 Discovery module 14

1.6 DDS Security 15

1.6.1 Authentication Service Plugin 15

1.6.2 Access Control Service Plugin 16

2 Vulnerabilità DDS **19**
2.1 Attacchi DDoS 19
2.1.1 DDoS blocco ricezione da parte del datareader Foglio 2 20
2.1.2 DDoS sfruttando estensione DDS security 21
2.2 Attacchi di enumerazione e di sniffing 22
2.2.1 Enumeration sniff foglio 2 e foglio 5 22
2.3 QoS Exploitation Attack foglio 1 24
2.3.1 Foglio 4-B Modifica maligna di ownership strength 25
2.3.2 Foglio 4-D Modifica maligna di LIFESPAN QoS 26

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 e per capire il suo funzionamento. Questo ci sarà utile per comprendere le vulnerabilità che verranno analizzate nei successivi capitoli. Inoltre verrà introdotta la sua estensione DDS security che si occupa di rendere più sicuro lo standard DDS aggiungendo l'autenticazione e una implementazione della cifratura dei pacchetti scambiati tra i vari dispositivi connessi alla rete. Infine verra mostrato in quali contesti attuali il DDS viene utilizzato con le sue diverse implementazioni.

In questa tesi potrebbe venire omessa la specifica OMG perchè faremo riferimento esclusivamente al DDS conforme all'Object Management Group. Questo standard ha delle specifiche tecniche ben precise, consentendo l'interoperabilità tra i diversi vendor che lo rispettano, insieme a tanti altri numerosi vantaggi [4].

1.1 Modello publish/subscribe

Prima di parlare del DDS, é necessario capire il funzionamento del modello Publish-Subscribe che sta alla base del suo funzionamento. Questo modello 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 avverrá 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 é quello di trasferire i dati dal sensore alla workstation in modo tale da poterli visualizzare a schermo. La metodologia piú frequente é quella di utilizzare un socket tramite protocollo TCP/IP, ma non sempre questa é la soluzione migliore. Alcuni vantaggi del TCP/IP sono la disponibilitá di utilizzo in molte applicazioni e nella maggior parte delle connessioni a Internet. Tuttavia, in certe implementazioni, TCP/IP non risulta la soluzione migliore, specialmente quando dobbiamo collegare un numero di dispositivi al network che può variare nel tempo. Se nel nostro network tra workstation e sensore della temperatura, aggiungiamo un altro dispositivo, come un sensore per la temperatura, bisognerebbe creare un nuovo socket TCP/IP per far comunicare il nuovo sensore con la workstation. Questo succede perchè TCP/IP supportá una comunicazione di tipo one-to-one (uno a uno). Come vedremó nella prossima sottosezione, il modello publish/subscribe non ha questa limitazione [14].

1.1.2 Struttura modello publish/subscribe

Per funzionare, il modello publish/subscribe, ha bisogno di due elementi chiamati publisher e subscriber, che permettono lo scambio di messaggi all'interno del network. La comunicazione avviene quando hanno un topic, che rappresenta una tipologia di dati, (ad esempio temperatura, distanza, velocitá, 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: è colui che si iscrive ai topic del publisher, cominciando cosí a ricevere nuovi dati sul topic scelto. Spesso si tratta di un dispositivo utilizzato per visualizzare informazioni, come uno schermo.

Chiamiamo entità tutti i dispositivi che fanno parte del modello publish/subscribe. Una caratteristica di queste entitá é che possono essere aggiunte o rimosse da una rete

senza nessun problema, dato che le comunicazioni avvengono in modalitá asincrona [4].

Inoltre questo modello risulta molto flessibile e adatto in ambienti real-time dove le informazioni possono cambiare o essere utilizzate da piú dispostivi. I subscriber, ad esempio possono cambiare i topic a cui sono iscritti e i publisher possono smettere di pubblicare nuovi aggiornamenti su un determinato topic anche a runtime [7].

1.2 Che cos'é il Data Distribution Service

Il DDS gestito da OMG è un middleware e uno standard API con una gestione dei dati di tipo data-centric. Prendendo in considerazione il modello ISO OSI (Open Systems Interconnection), questo middleware è un software che si trova tra l'applicativo e il livello di sistema operativo. Il DDS si basa sul DCPS (Data-Centric Publish-Subscribe) che é un modello di comunicazione simile a quello 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. Con data-centric infatti, specifichiamo che il focus del modello DCPS è incentrato sui dati stessi, anzichè sulle entitá che li scambiano [9]. Così facendo non bisogna preoccuparsi dell'invio o della ricezione dei messaggi, perchè questa parte viene completamente gestita dal middleware.

Altri vantaggi del DDS includono una architettura adattabile che supportá degli elementi di auto-scoperta (auto-discovery) o Dynamic Discovery in modo tale da aggiungere o rimuovere dispositivi dalla rete in modo automatico, anche a runtime. Il Dynamic Discovery ha il compito di analizzare quali tipologie di dati ha bisogno questo nuovo dispositivo.

Inoltre ogni nuovo partecipante utilizzerà sempre le stesse API per comunicare con l'applicativo dato che non c'é bisogno di configurare le impostazioni degli indirizzi IP o preoccuparsi delle diverse architetture [14].

1.2.1 Quality of Service (QoS)

Per soddisfare i diversi requisiti di una trasmissione dati, il Data Distribution Service (DDS) utilizza un insieme di policy Quality of Service (QoS). Queste policy permet-

tono di controllare, regolare e ottimizzare lo scambio di dati tra i vari componenti all'interno del middleware. Queste 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, in modo tale da addattarsi alle esigenze dell'applicazione.

1.2.2 Global Space Data

Il DDS utilizza il Global Data Space (spazio dati globale), un'area logica condivisa che consente agli applicativi di accedere a una sorta di memoria locale tramite API. L'applicativo nella scrittura o nella ricezione dei dati utilizzerá questa memoria locale fittizia come un'unica risorsa centralizzata. Tuttavia, i dati all'interno di questa memoria possono contenere informazioni provenienti da nodi remoti distribuiti per la rete. L'applicativo non deve cosi preoccuparsi dell'accessibilitá dei dati, poiché questi vengono gestiti come se si trovassero tutti in unico punto [9].

1.2.3 Architettura DDS

Lo standard DDS definito dall'OMG è composto da due layer: il DDS e il DDSI (DDS Interoperability).

- DDS: é il layer fondamentale in cui troviamo il DCPS (Data-Centric Publish-Subscribe), il modello di comunicazione simile a quello di tipo publish/subscribe, che si occupa di mettere in comuncazione piú applicazioni tra di loro. In questo layer vengono inoltre definite le policies QoS [2].

- DDSI: é il layer che si occupa di garantire l'interoperabilità tra le diverse implementazioni del DDS, ad esempio quando provengono da vendors diversi. All'interno di questo layer troviamo l'RTPS (Real-Time Publish-Subscribe Protocol), un protocollo che permette ai vari dispositivi DDS di comunicare e scoprirsi tra di loro (Dynamic Discovery). RTPS é il wire-protocol (un protocollo che permette lo scambio di messaggi del DDS al layer di trasporto di rete del modello ISO OSI) ufficiale del DDS, con standard definito da OMG, che definisce il formato dei messaggi e impone le regole che permettono una trasmissione

standardizzata di scambio dati. Se questo wire-protocol non fosse presente, le diverse implementazioni del DDS non potrebbero comunicare tra di loro [10].

1.3 Le entitá del DCPS

Il layer DDS per operare utilizza le entitá definite dal DCPS, che rappresentano gli elementi necessari per il funzionamento dell'intero middleware. Queste hanno il compito di gestire i dati scambiati tra i vari partecipanti all'interno del sistema. Le entitá 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 entità deve tener conto del suo set di policies QoS configurate che ne definiscono il comportamento. Queste policy verranno analizzate piú nel dettaglio in una prossima sezione.

Il linguaggio utilizzato da queste entitá si chiama Interface Definition Language(IDL) che anch'esso è standardizzato da OMG. IDL è un linguaggio tipizzato simile a C++ che supportá data types come char, int, double, float etc [14] ...

1.3.1 Publisher e Subscriber

Publisher e subscriber sono entità che abbiamo giá incontrato in precedenza nel modello publish/subscribe. Queste due entitá mantengono lo stesso ruolo all'interno del modello DCPS.

1.3.2 DataWriter e DataReader

Per comunicare tra loro, un publisher si interfaccia tramite DataWriter, mentre il subscriber si interfaccia tramite un DataReader. Il DataWriter e il DataReader sono interfacce, perché l'applicativo può mandare e ricevere dati tramite queste due entitá fondamentali. Questi dati scambiati tra middleware e applicativo durante le comunicazioni sono i data type e i data-value. I data type consentono di descrivere la struttura e il formato del dato, mentre i data-value sono i dati veri e propri che rispettano le specifiche data type.

- DataWriter: é l'interfaccia usata dagli applicativi per comunicare i data-values con un loro specifico data type ai publisher. Ricevuti questi data-values il publisher spedirá le informazioni ricevute dall'applicativo, ai relativi subscriber.

- DataReader: é l'interfaccia usata dagli applicativi per ricevere i data-values con i rispettivi data type pubblicati in precedenza da un publisher [4].

É possibile associare piú DataReaders a un unico publisher e piú DataWriters a un unico subscriber. Tuttavia, un DataWriter può essere associato solo a un Publisher e un DataReader può essere associato solo a un Subscriber. Questo avviene perchè i DataReaders e DataWriters possono utilizzare un solo data type alla volta, mentre i publiishers e i subscribers, non avendo questa limitazione, possono gestire piú DataWriters e DataReaders alla volta.

1.3.3 Topic

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

I topic vengono utilizzati per identificare il data type scambiato tra i publishers e i subscribers, creando cosi un punto di connessione tra DataWriter e DataReader [8]. All'interno del topic troviamo il nome, i data types e le policy QoS. Il nome del topic corrisponde ad una stringa univoca dato che serve per identificarlo tra gli altri topic del domain.

1.3.4 Key e Instanza

Uno o piú data types di un topic, possono diventare la chiave (key) del topic. Queste chiavi ci permettono di suddividere i data-values di un singolo topic, dividendoli a seconda della chiave. Ogni suddivisione che effettuiamo tramite una key diversa crea un'istanza. L'istanza, quindi al suo interno contiene i data-values di un flusso di data (data-stream) [11].

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 che identifica da quale vettura i dati provengono.

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

}

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

Creiamo ora due istanze, una con valore 270 per velocità e con l'id-macchina uguale 1 e l'altra con un valore di 220 per la velocita e con id-macchina uguale a 2. In questo caso l'id-macchina 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'entitá del domain (dominio) rapresenta uno spazio logico definito con lo scopo di mettere in comunicazione i vari applicativi tra di loro. All'interno possiamo trovare i vari topic che collegano gli applicativi con i loro rispettivi data-types. L'entitá domain é caratterizzato dalle seguenti proprietá:

- Ogni domain viene identificato da un id per renderlo univoco.
- Ogni entitá del DDS puó appartenere a un solo domain.
- Le entitá all'interno del domain possono interagire solamente con le altre entitá all'interno dello stesso domain
- Due applicativi DDS per poter comunicare tra di loro hanno bisogno di entrare nello stesso domain.
- Un applicativo puó far parte di piú domain creando piú istanze dell'entita DomainParticipant in ogni domain in cui vuole iteragire [13].

1.3.6 DomainParticipant

L'entitá del DomainParticipant viene utilizzata dagli applicativi per entrare all'interno di un Domain del DDS. Rappresenta la prima entitá creata da un'applicazione che verrá impiegata per creare DataWriters e DataReaders al suo interno. Ha il compito di inizializzare le comunicazioni con il Domain dove si trova attraverso il processo di discovery. Questo processo consente alle entitá appartenenti allo stesso domain di

trovarsi e connettersi automaticamente. L'entitá DomainParticipant é caratterizzato dalle seguenti proprietá:

- Il DomainParticipant come le altre entitá, puó esistere solo all'interno di un dominio.
- Il DomainParticipant é responsabile della scoperta di altri DomainParticipant all'interno del domain [12].

1.4 Le policy QoS nel dettaglio

A livello logico, le specifiche del DDS definiscono un insieme di polices QoS che le entitá del DCPS devono rispettare. Qui di seguito vengono proposte le categorie QoS piú rilevanti.

- Ownership: questo valore specifica se un topic puó essere aggiornato da piú (SHARED ownership) o un solo (EXCLUSIVE ownership) publisher. Se abbiamo impostato un ownership di tipo EXCLUSIVE, per decidere il publisher che ha la possibilità di aggiornare il topic, viene utilizzato l'ownership STRENGTH.
- Liveness: viene utilizzato per specificare se é necessaria una comunicazione di tipo attivo, rispetto ad una di tipo intermittente.
- Reliability: specifica se in una comunicazione tutti i dati trasferiti tra publisher e subscriber devono essere consegnati per intero, oppure se é accettabile anche la perdita di alcuni dati.
- Lifespan: specifica il tempo di scadenza dei dati pubblicati da un publisher.
- History: specifica quanti e come i dati devono essere mantenuti in un subscriber dopo averli ricevuti.
- Durability: specifica se i dati inviati in precedenza sono disponibili per i nuovi subscriber appena entrati nel domain.

Le configurazioni delle policy vengono trasmesse alle varie entitá dal DomainParticipant come i publisher, i subscriber, i topic, i DataWriters etc... Tuttavia, le policy di un publisher e un subscriber devono essere compatibili. Se cosí non fosse, la comuncazione tra i due potrebbe essere compromessa [2].

1.5 Il protocollo RTPS

RTPS (Real-Time Publish-Subscribe Protocol) é il wire-protocol nativo utilizzato dal DDS. Esso consente di trasferire i dati provenienti dal layer DDS a quello di trasporto della rete. Solitamente viene utilizzato in combinazione con il protocollo best-effort UDP/IP, che risulta ottimale per le comunicazioni di tipo real-time, tuttavia protocolli connection-oriented come il TCP/IP possono essere utilizzati. RTPS include molti vantaggi ideali per il DDS:

- Connettivitá plug and play: le nuove applicazioni possono unirsi o lasciare il domain a proprio piacimento.
- Tolleranza ai guasti: non sono presenti singoli punti di guasto perchè i dati vengono distribuiti e replicati tra le varie entitá DDS che adoperano il Global Data Space.
- Type-safety: gli errori di programmazione vengono gestiti in modo tale da non compromettere il funzionamento dei dispositivi remoti.

L'RTPS é suddiviso in quattro moduli differenti: lo structure module, il messages module, behavior module e il discovery module [6].

1.5.1 Structure module

Questo modulo si occupa di associare le entitá DDS alle corrispondenti entitá RTPS. Queste entitá RTPS sono utilizzate per rappresentare le entitá del DDS (come DataReader e DataWriter) all'interno del protocollo RTPS [6].

1.5.2 Behavior module

Il behavior module definisce delle regole di comunicazione durante una seguenza di messaggi tra due o piú entita RTPS, che sono i RTPS Writers e i RTPS Readers. Queste regole servono a mantenere l'interoperabilità tra le varie implementazioni (anche di diversi vendors) del DDS [6].

1.5.3 Messages module

Il messages module si occupa di descrivere il formato dei messaggi scambiati tra i RTPS Writers e i RTPS Readers. Questi messaggi RTPS sono composti da un header seguito da dei sottomessaggi. Nell'header troviamo informazioni relative al protocollo RTPS, come la sua versione, il nome del vendor dell'implementazione usata e il mittente. Nel sottomessaggi invece possiamo trovare un header e una serie elementi. Nell'header del sottomessaggio é presente l'id che identifica il tipo di sottomessaggio, eventuali flag e la lunghezza in bytes del sottomessaggio. Le tipologie di sottomessaggi piú importanti, identificate dal submessage header, sono:

- DATA: in questo sottomessaggio vengono trasferiti dall'RTPS Writers all'RTPS Reader i dati effettivi relativi ad un topic.
- HEARTBEAT: viene mandato da un RTPS Writer a un RTPS Reader per comunicare il numero di nuovi (sequence number) aggiornamenti che il Writer ha disponibili.
- ACKNACK: utilizzato per comunicare lo stato di un RTPS Reader al corrispondente RTPS Writer, per informarlo riguardo i dati ricevuti e quelli mancanti. Questo sottomessaggio con il con la flag FINAL impostata consente di far rimanere il Reader sincronizzato con Writer [6].

1.5.4 Discovery module

Questo modulo garantisce che i nuovi partecipanti DDS (publisher e subscriber) riescano a identificarsi in automatico tra di loro in modo tale da inizializzare una possibile comunicazione. Questo modulo é responsabile del Dynamic Discovery delle entitá del DDS all'interno dello stesso domain. Il Dynamic Discovery utilizza messaggi di tipo multicast e unicast per informare gli altri partecipanti di un nuovo dispositivo connesso alla rete, pronto cominciare le comunicazioni dal resto delle entitá. Il discovery module é composto da due protocolli chiamati Simple Participant Discovery Protocol (SPDP) e Simple Endpoint Discovery Protocol (SEDP). SPDP ha il compito di scoprire nuovi partecipanti, mentre l'SEPD si occupa di scambiare tra le entitá le informazioni di topics, DataWriter e DataReader. In particolare l'SEDP serve per collegare tramite topic i DataReaders ai DataWriters [6].

1.6 DDS Security

Nelle specifiche del DDS non vien presa in considerazione la sicurezza, quindi un'implementazione che utilizza il DDS di base può essere esposta a numerosi a rischi. Per ovviare a questo problema, OMG ha definito un nuovo standard chiamato DDS security. Il DDS security è un estensione del DDS con l'obiettivo di mitigare una moltitudine di vettori d'attacco come la lettura e la scrittura dei messaggi scambiati tra i partecipanti di un domain DDS. Questa estensione è composta da cinque plugin:

- Authentication Service Plugin: serve per effettuare l'autenticazione delle entitá DDS. Senza l'autenticazione le enitá non possono comunicare tra di loro.
- Access Control Service Plugin: ha lo scopo di imporre delle policies alle entitá DDS autenticate. Ad esempio limitare la pubblicazione di nuovi dati o la creazione di nuovi topic.
- Cryptographic Service Plugin: gestisce tutte le operazioni crittografiche, come la crittografia, la decrittazione e le firme digitali. Ha anche il compito di controllare l'integrità dei messaggi.
- Logging Service Plugin: permette di effettuare un audit di tutte le operazioni DDS rilevanti all'interno di un domain.
- Data Tagging Service Plugin: fornisce dei metodi per implentare un tag su tutti i dati trasferiti.

Anche se il DDS security riesce a risolvere molti problemi legati alla sicurezza del DDS, non sempre è possbile utilizzarlo. Spesso la sua configurazione può richiedere molto tempo per essere impostata, soprattutto su sistemi DDS già esistenti e sprovvisti di questa estensione. Il partecipante con la sicurezza più debole rappresenta la sicurezza complessiva dell'intero sistema, quindi ogni entitá deve essere protetta [2].

Il DDS security può essere usato anche non implementando tutti i plugin; gli ultimi due sono facoltativi e vengono raramente usati [10].

1.6.1 Authentication Service Plugin

Senza questo modulo chiunque potrebbe entrare a far parte del domain del DDS, ponendo un grave rischio per la sicurezza. Per ovviare a questa falla, l'Authentication

Service Plugin consente, a ogni dispositivo che vuole entrare del domain DDS, la possibilità di autenticarsi. Prima di effettuare l'autenticazione tutti i partecipanti devono configuare un loro certificato, le loro chiavi private, mentre l'amministratore deve creare un certificato root(o Certificate Authority, CA) che deve essere riconosciuto da tutte le entitá autorizzate.

Processo di autenticazione

L'autenticazione di due entitá in modo sicuro viene effettuata tramite il protocollo Diffie-Hellman, che consente la trasmissione di chiavi in modo sicuro anche in canali di comunicazione non sicuri. Le due entità utilizzando Diffie-Hellman otterranno una chiave segreta condivisa da entrambi. Implementando questo protocollo la chiave non verrá mai trasmessa direttamente, evitando cosí di essere intercettata da possibili attaccanti.

Per rafforzare ulteriormente la sicurezza e prevenire attacchi replay (riutilizzo di messaggi intercettati) o di impersonificazione, vengono utilizzate le challenge. Queste challenge corrispondo a valori casuali che cambiano nel tempo e vengono utilizzati durante il calcolo della firma digitale che si effettua nel protocollo Diffie-Hellman. Dato che che queste challenge cambiano periodicamente, i vecchi messaggi intercettati dagli attaccanti non possono essere più riutilizzati.

La fase di autentincazione si conclude quando viene Completato lo scambio delle chiavi. Queste chiavi verranno poi utilizzate da protocolli di crittazione come l'RSA (Rivest-Shamir-Adleman) per effettuare comunicazioni in modo sicuro. Infatti non sarà possibile per un attaccante spiare o cambiare il contenuto dei messaggi dato che quest'ultimi sono criptati [15].

1.6.2 Access Control Service Plugin

Questo plugin gestisce i permessi delle entitá all'interno di un domain DDS. É possibile configurare questi permessi con una granularitá molto fine. Dei possibili permessi possono essere: aggiornare un determinato topic da parte di un DataWriter, far entrare una determinata entitá all'interno di un domain, eliminare un topic, entrare o uscire da determinati domains etc...

L'Access Control Service Plugin per funzionare ha bisogno di due file in formato XML che devono essere entrambi firmati da un CA: il governance document e il permissions document. Il governance document rimane uguale per tutti i dispositivi all'interno del domain DDS e si occupa di gestire permessi generali a livello di domain. Il permissions document, invece, é unico per ogni dispositivo e si occupa di gestire i permessi del singolo partecipante. Questi vengono ricevuti dai partecipanti durante la fase di autenticazione e devono rimanere sempre disponibili [10].

```
...
    <permissions>
        <grant name="ShapesPermission">
            <subject_name>CN=DDS Shapes Demo</subject_name>
            <validity>
                <not_before>2013-10-26T00:00:00</not_before>
                <not_after>2018-10-26T22:45:30</not_after>
            </validity>
            <allow_rule>
                <domains>
                    <id>0</id>
                </domains>
            </allow_rule>
            <deny_rule>
                <domains>
                    <id>0</id>
                </domains>
                <publish>
                    <topics>
                        <topic>Circle1</topic>
                    </topics>
                </publish>
...
```

Codice 1.2: Estratto di permissions document, tratto da documento di riferimento del DDS Security versione 1.1 [5].

Processo di controllo permessi

Un esempio di processo di controllo access accade quando un DataWriter richiede l'autorizzazione per creare un nuovo topic. Le altre identitá hanno il compito di verificare se l'operazione richiesta dal DataWriter é consentitá dai i files di permessi a loro disposizione. Per effettuare questa operazione il partecipante che richiede l'autorizzazione deve mandare il proprio permission document, il topic che intende creare e i propri metadati. Successivamente un altro partecipante, che ha il compito di autorizzare il DataWriter, riceverá il messaggio ed effettuerá le seguenti verifiche:

1. Verifica che la firma digitale del permesso ricevuto sia valida.
2. I metadati forniti devono corrispondere a quelli del del permesso ricevuto.

Il secondo controllo ha lo scopo di verificare che il partecipante che vuole creare il nuovo topic sia effettivamente quello indicato nel permesso ricevuto in precedenza; mitigando in questo modo gli attacchi di impersonificazione [15].

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.[16]

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

[16]

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 numember 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.[16]

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

[5] Un'entitá deve effettuare l'autenticazione nel caso in cui apre una nuova comunicazione con una nuova entità prima sconosciuta. Queste due entitá prima effettuare la comunicazione devono scambiarsi le chiavi in modo sicuro tramite Diffie-Hellman in modo tale da poter crittare i successivi messaggi. Durante l'utilizzo del protocollo Diffie-Hellman vengono utilizzate le challenge, che sono valori che variano nel tempo, per calcolare la firma digitale richiesta dal protocollo [15].

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'entità con cui vuole aprire una comunicazione. 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 questi calcoli crittografici 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.[15]

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. [15]

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.[16]

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 i messaggi scambaiti sono molto "verbose", cioè scambiano

informazioni in chiaro durante le comunicazioni tra i vari dispositivi.[16] 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. [6]

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.[16]

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

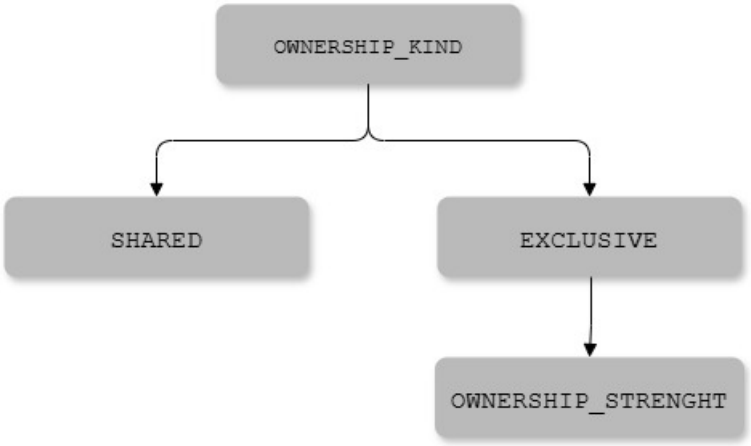

Figura 2.1: Illustrazione policy QoS del DDS

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.

[4]

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.[3]

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. [3]

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. [3]

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 a riesce 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. [3]

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 $<$ 80ms 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 >= 120ms 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 < 80ms utilizzando il la RELIABILITY di tipo BEST-EFFORT, per ottenere lo stesso risultato con RELIABILITY di tipo RELIABLE dobbiamo impostare un LIFESPAN < 0.8ms. [3]

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. [3]

Tipo di attacco	**Vettore attacco**	**Protoc./ Estens.**	**Bersaglio nella rete**	**Software**	**Soluzione**
Discovery devices[16]	Verbose nature of RTPS	DDSI-RTPS	Tutti i partecipanti	Sniffer python	WireGuard
DDos[16]	Heartbeat sequence number	DDSI-RTPS	DataReader	Sniffer python	-
DDoS[15]	Authentication challenge	DDS security 1.1 Discovery protoc.	Tutti i partecipanti	Proverif	Scadenza richieste di autenticazione
QoS policy[3]	ownership-strength	DDSI-RTPS	DataReader	RTI shapes	DDS security
QoS policy[3]	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 24

Elenco delle tabelle

2.1 La versione DDS in tutti i casi è la 1.4 27

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. MALICIOUS USE OF OMG DATA DISTRIBUTION SERVICE (DDS) IN REAL-TIME MISSION CRITICAL DISTRIBUTED SYSTEMS, April 2017. URL `https://espace.rmc-cmr.ca/jspui/handle/11264/1241`. [Online; accessed 11. Feb. 2025].

[3] 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`.

[4] Object Management Group. OMG Data Distribution Service, April 2015. URL `http://www.omg.org/spec/DDS/1.4/PDF`. [Accesso: 2 febbraio 2025].

[5] Object Management Group. DDS Security, July 2018. URL `https://www.omg.org/spec/DDS-SECURITY/1.1/PDF`. [Accesso: 2 febbraio 2025].

[6] Object Management Group. About the DDS Interoperability Wire Protocol Specification Version 2.5, February 2025. URL `https://www.omg.org/spec/DDSI-RTPS/2.5/About-DDSI-RTPS`. [Online; accessed 12. Feb. 2025].

[7] 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.

[8] OMG. Topic [DDS Foundation Wiki], February 2025. URL https://www.omgwiki.org/ddsf/doku.php?id=ddsf:public:guidebook:06_append:glossary:t:topic. [Online; accessed 5. Feb. 2025].

[9] OMG. What is DDS?, February 2025. URL https://www.dds-foundation.org/what-is-dds-3. [Online; accessed 6. Feb. 2025].

[10] J.J. Rengers. Dds in a zero trust cloud native environment in the naval domain, November 2022. URL http://essay.utwente.nl/93639/.

[11] RTI. Instance | data distribution service (dds) community rti connext users, February 2025. URL https://community.rti.com/glossary/instance. [Online; accessed 2025-02-05].

[12] RTI. DomainParticipant | Data Distribution Service (DDS) Community RTI Connext Users, February 2025. URL https://community.rti.com/glossary-term/domainparticipant. [Online; accessed 6. Feb. 2025].

[13] RTI. Domain | Data Distribution Service (DDS) Community RTI Connext Users, February 2025. URL https://community.rti.com/glossary/domain. [Online; accessed 6. Feb. 2025].

[14] 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.

[15] Binghan 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.

[16] 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 incidunt 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