digital whisper Technologic Papers

Secure DDS: Securing Realtime Data Like a Boss

מאת יותם דוד ונתנאל כהן

הקדמה

תהליכים לקבלת אינפורמציה מאפיינים את חיינו מכל עבר. בין אם מדובר בשאלה "כיצד אתה מרגיש?" בביקור השיגרתי אצל הרופא ובין אם מדובר בפנייה לממשק באתר הבנק שמבצע בתורו קריאה לשרת ביבשת אחרת כדי שנוכל להתעדכן בשווי מנייה בשוק ההון העולמי. והרי שאין זה פלא, מידע הוא הכלי החזק ביותר שיש לנו בקבלת החלטות בצורה רציונאלית. אינספור אלגוריתמים, פרוטוקולים ומערכות מחשוב נכתבו במטרה לקבל, לפרסר ולעבד כמויות גדולות של מידע בשביל לְצַיֵּד את משתמשהן בזמן הקצר ביותר בתשובה המדויקת ביותר.

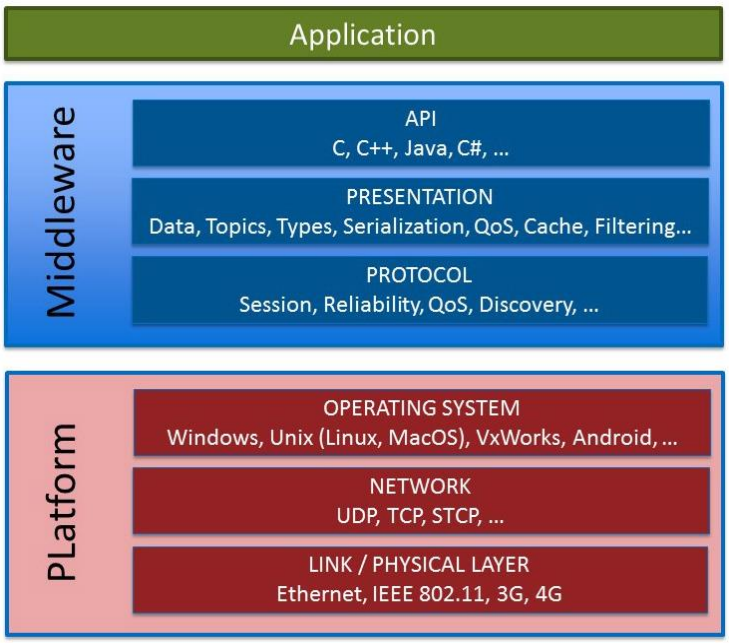
אבל לא כל התהליכים נולדו שווים, בעוד שאופי המידע המאוחסן הינו סטטי מטבעו, תהליך הבאתו מוגדר בדינאמיות וצורת מימושו שונה משמעותית בין מקום למקום. אם שנייה או שתיים של המתנה לטעינת אתר הבנק לא תהרוג אתכם, ובכן, עיכוב זהה במערכת הבלמים ברכב כנראה שכן. מסיבה זו, כשניגשים לאפיין תהליך, חייבים בראש ובראשונה להגדיר מאפיין אחד - דחיפות קבלת המידע. מסיבה זו פרוטוקול הזרמת הנתונים מקבל חשיבות מכרעת. על מערכות מחשוב העונות על צורך המיידי של צריכת מידע ועיבודו נאמר כי הן "**Real Time**" ואכן פרוטוקולים כדוגמת XMPP, DDS, WebRTC, MQTT ורבים נוספים תוכננו בשביל לתת מענה לשליחת וקבלת מידע על פני מגוון רחב של תעשיות, שימושים ואפליקציות.

מתוך הנחת היסוד הנפוצה שאבטחה מגיעה בחשבון ישיר על תפעוליות, אלמנט שלא פעם נשאר מאחור כאשר דנים בסוגיות מערכות זמן אמת הוא נושא הגנת המידע. ובאמת ישנם לא מעט פרוטוקולי Real Time שמתעלמים באלגנטיות מסוגיות בסיסיות כמו הצפנה חזקה ואימות.

במאמר הקרוב אנחנו הולכים להציג כיצד הפרוטוקול **DDS** (קיצור של Data Distribution Service) מתמודד מול סט האיומים האבטחתיים אשר אליו הוא חשוף וכיצד באמצעותו ניתן להגן בצורה איכותית על המערכות המשתמשות בו.

אבל למה דווקא DDS ולמה לכם בכלל לקורא מאמר שלם עליו? ובכן, מעצם היותו פרוטוקול Middleware שנמצא בין שכבת מערכת ההפעלה לשכבת האפליקציה הוא מקנה אלסטיות רבה למי שעובד איתו. קיימים מימושים רבים לפרוטוקול המכילים APIs במגוון רחב של שפות כגון: C, C#, C++, Java, Paython ועוד. התמיכה הרחבה הזו היא שמאפשרת לפרוטוקול לתמוך בתקשורת בין מספר רב של מערכות בעלות פלטפורמות שונות ומגוונות. ישנם מימושי Open Source חינמיות כגון Fast DDS ומימושים מבוססי רישיונות כגון RTI DDS.

DDS הינו סטנדרט תוכנתי לשליחה והחלפה של מידע בזמן אמת. DDS מתוכנן להעביר את המידע בצורה מהירה ואמינה בין אפליקציות, מכשירים ומערכות מבוזרות.



[מקור: https://www.dds-foundation.org/what-is-dds-3]

ממאפייניו הבולטים:

- מודל **Publish-Subscribe** - רכיבים או אפליקציות שמוגדרים כ-Publishers מסוגלים לכתוב מידע לנושאים קיימים, בעוד ש-Subscribers מסוגלים לקרוא מידע מנושאים מסוימים.
- DDS - **Quality of Service** מספק מגוון רחב של הגדרות QoS המאפשרים למפתחים להגדיר כיצד המידע יאומת, באילו קבועי זמן המידע ישלח, באילו משאבי מערכת להשתמש ועוד.
- סקלאביליות - הפרוטוקול יודע להתנהל עם כמויות גדולות של מידע ומספר רב של רכיבים ברשת.
- אבטחה - אימות מידע, אימות רכיבים ברשת, הצפנת, הגבלת גישה ועוד. כמובן שניגע ביכולות הללו בהרחבה במאמר זה. חשוב כבר עכשיו לציין כי יכולות ההגנה הללו אינן ממומשות ב-DDS כברירת מחדל, אלא ניתן להוסיף אותם כ-Plugins נפרדים.
- תמיכה רחבה בפלטפורמות DDS - תומך במגוון רחב של מערכות הפעלה, חומרות ושפות תכנות.

היכן נשתמש ב-DDS?

אפשר לראות לא מעט את ה-DDS בתעשיות בהן תקשורת והפצת מידע בזמן אמת הן הכרחיות וקריטיות ביותר. הפרוטוקול אומנם אינו פופולרי בחברות ההייטק, אך השימוש בו נוגע בנו בחיי היום יום יותר ממה שאנחנו חושבים. הדוגמאות הבולטות לכך כיום בעולם הינן:

- רכבים אוטונומיים - בתעשיית הרכבים האוטונומיים, פרוטוקול זה בשימוש רחב בשל הצורך להעביר מידע בין הסנסורים השונים, מצלמות, מערכות השליטה והאלגוריתמים שצריכים לבצע ולקבל החלטות בזמן אמת. הפרוטוקול תומך בפונקציות כגון: זיהוי אובייקטים, ניווט להימנעות מהתנגשויות. חברות בולטות שמשתמשות בפרוטוקול הינן Waymo ו-Volkswagen.

- אנרגיה חכמה - תעשיות העוסקות בייצור של אנרגיה חכמה בעזרת טורבינות רוח, פאנלים סולריים ואפילו כורים גרעיניים משתמשים בפרוטוקול לשם בקרה, חלוקת עומסים ואבטחה.
- מערכות הגנה וכלי טייס - באפליקציות ומערכות בתחום התעופה, החלל ובתעשיות ביטחוניות, הפרוטוקול בשימוש רחב בשל הסקלאביליות והתאימות שהוא מספק. הוא מאפשר להעביר מידע של סנסורים, רדארים, מצבי כוחות בשטח וכו' למרכזי הבקרה והשליטה. לדוגמה - **NASA** עושה שימוש ב-DDS בשביל להעביר מידע בין חליליות, טילים ותחנות חלל למרכז הבקרה. כן כן, קראתם נכון.

- מערכת הבריאות - קיים שימוש בפרוטוקול לזה לשינוע והעברה של מידע רפואי של מטופלים בזמן אמת, לדוגמה - ניטור מדדים רפואיים במוניטור, מכשירים כמו MRI ו-CT.
- סימולטורים ומאמנים - במרכזי אימון בצבא קיימים סימולטורים שמדמים תרחישים בזמן אמת, בהם קיים שימוש נרחב ב-DDS להעברת המידע בצורה אמינה ומדויקת ביותר.
- שירותים פיננסים - שירותים המאפשרים קנייה ומכירה של מניות בזמן אמת ובתדירות גבוהה, בהם הדיוק של זמן השליחה הינו קריטי.
- ערים חכמות DDS - נותן מענה לשליטה במרחב העירוני ע"י מתן גישה למידע בסנסורים הפרוסים ברחבי העיר ומספר יכולת ניתוח ובקרה עליהם. ישנן ערים שמשתמשות בפרוטוקול כדי לסנכרן ולחבר שירותי חניה בעיר ולעקוב אחר מצב הפקקים ולספק את המידע הזה בזמן אמת לפלאפונים של תושבים המעוניינים בכך. דוגמה בולטת לעיר חכמה שמשתמשת בפרוטוקול היא וינה באוסטריה.

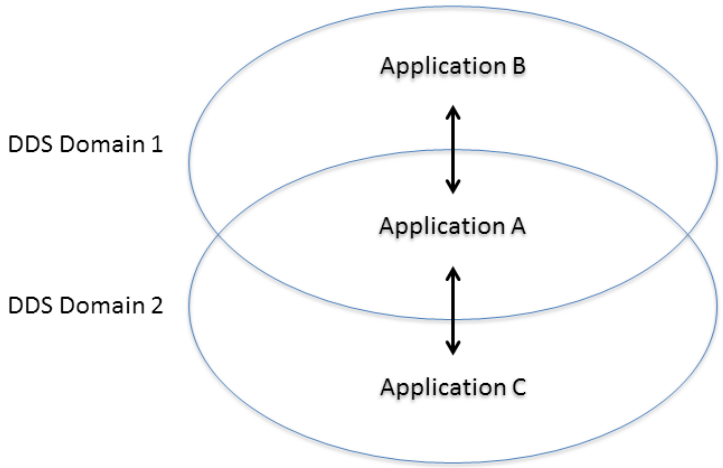
ארכיטקטורת DDS

לפני שנצלול ללב ההגנה של הפרוטוקול שווה לומר מילה או שתיים על התשתית ולכן נתחיל משתי הגדרות בסיסיות:

DDS Domain - זהו המרחב הלוגי בו נמצאות כלל האפליקציות המתקשרות ביניהן בפרוטוקול DDS והאובייקטים המנוהלים ביניהם, קרי, המידע שאפליקציות הללו כותבות או קוראות אחת מהשנייה.

DomainParticipants - כלל האפליקציות שנמצאות בתוך ה-DDS Domain. לאפליקציות הללו יש יכולת ליצור, לשנות, למחוק ולנהל אובייקטים בתוך ה-DDS Domain.

אפליקציות יכולות לתקשר אחת עם השנייה אך ורק אם הן באותו DDS Domain, כאשר אפליקציה יכולה להיות שותפה ביותר מ-DDS Domain אחד:



[מקור: https://community.rti.com/]

מהאיור ניתן לראות שאפליקציה A יכולה לדברי עם אפליקציות B ו-C אך אפליקציה B לא יכולה לדבר עם C ישירות. כפי שציינו קודם לכן, פרוטוקול DDS עובד במודל של **Publish-Subscribe**. כלומר, ישנם DomainParticipants שכותבים מידע, קרי Publishers בעוד שאפליקציות אחרות קוראות את המידע הזה, קרי Subscribers.

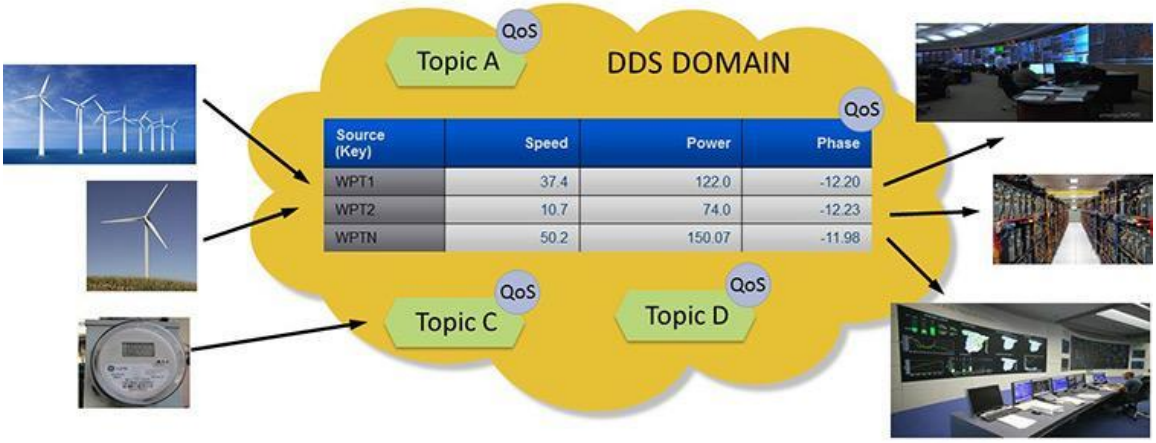
המידע שנשלח בתוך ה-DDS Domain נשמר במבנה לוגי שנקרא **Topic**. יחידה של מידע שנכתבת או נקראת מתוך ה-Topic נקראת **Sample**. כל Topic מכיל 3 מאפיינים עיקריים:

1. **Topic Name** - השם הינו מזהה ייחודי מסוג String בו האפליקציות משתמשות כדי לתאר איזה מידע הן רוצות לכתוב או לקרוא. לדוגמא: "מהירות רכב".
2. **Data Type** - המבנה והפורמט של המידע שנכתב ונקרא מה-Topic. בהמשך לדוגמא הקודמת, אם ה-Topic שלנו הוא "מהירות רכב", המידע שנשמר בו יכול להיות בעל המבנה הבא:

```
struct CarSpeed {
        string licensePlate;  // License plate number of the car
        float speed;          // The car's current speed in miles per hour
        long timestamp;       // Timestamp when the speed reading was recorded
};
```

3. **QoS Settings** - לכל Topic יש הגדרות Quality of Service המאפיינות את התקשורת הקשורה אליו. ניתן להגדיר לדוגמא: מהו גודל הודעה מקסימלי, איזה מידע תחת ה-Topic צריך להיות אמין, איזו היסטוריית הודעות צריך לשמור וכו'.

התמונה הבאה ממחישה את כל המושגים שהסברנו עד כה - ישנם מספר מכשירים בתחום הנפקדת האנרגיה, שמתפקדים ב-DDS Domain כ-Publishers וכותבים Samples המכילים מאפיינים כגון Speed, Power, Phase אל תוך ה-Topic. בנוסף אליהם קיימים ב-Domain מספר מערכות ומרכזים שמתפקדים כ-Subscribers וקוראים את המידע הזה מתוך ה-Topic:



[מקור: https://www.dds-foundation.org/what-is-dds-3]

חשוב להדגיש - אמנם לאפליקציות עצמן זה נראה שהן נגישות לזיכרון מקומי דרך ה-API בו הן משתמשות, אך כל המידע שנשלח ב-DDS Domain מבוזר בין כל ה-DomainParticipants. אין באמת "מקום אחסון מרכזי אחד" וזאת בשביל למנוע מצב של נקודת כשל מרכזית ברשת. לכן, בפועל כאשר אפליקציה כותבת Sample, היא שולחת את המידע ב-Multicast או Unicast ל-Subscribers המתאימים.

DDS vs RTPS

אימפלמנטציה של שכבות לוגיות שונות בפרוטוקול היא פרקטיקה ידועה המאפשרת למפתחים אלסטיות רבה כאשר הם באים לעבוד ולהתממשק עם הפרוטוקול למוצרים אחרים. **RTPS** מהווה בדיוק את זה.

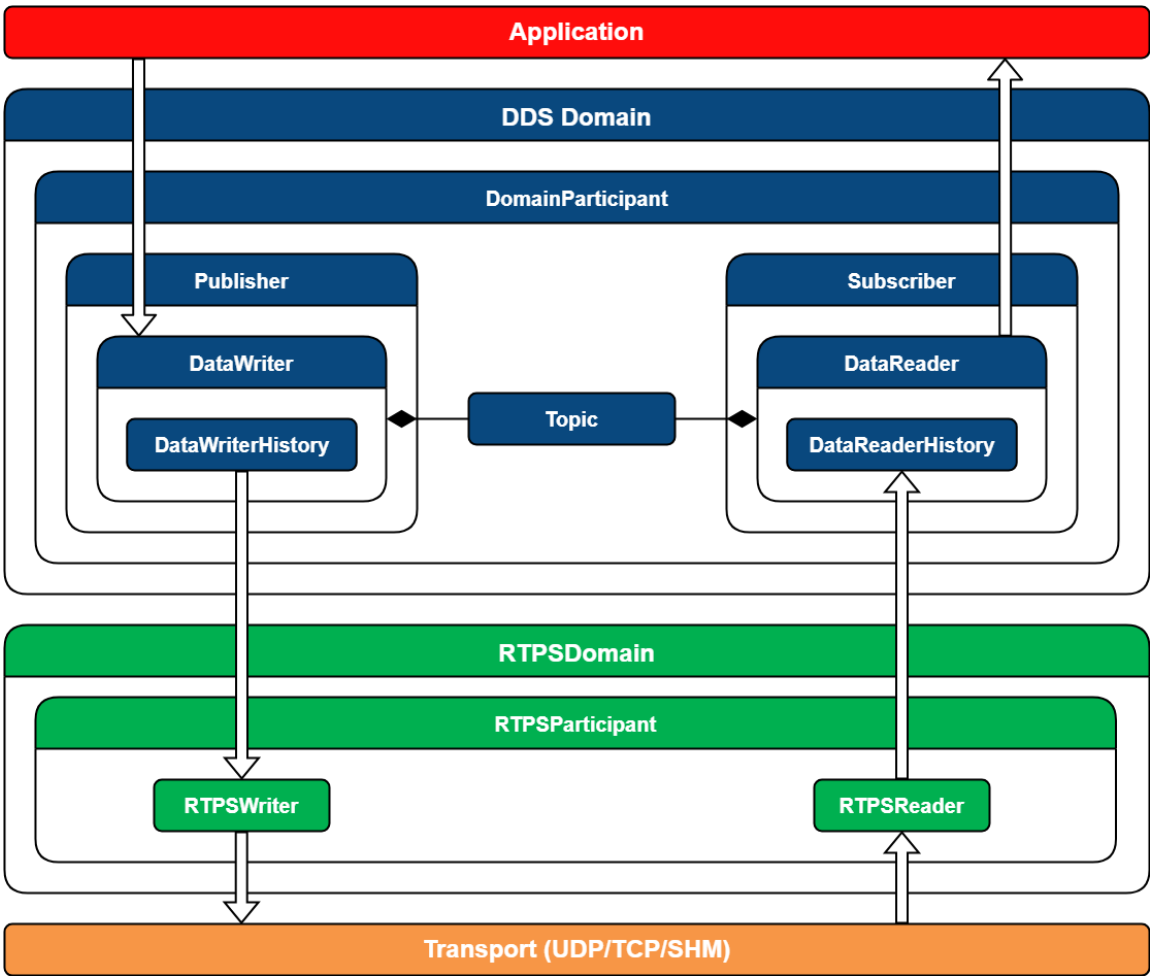
RTPS (קיצור של Real Time Publish Subscribe) מהווה את שכבת האבסטרקציה שבין פרוטוקול DDS לשכבת התעבורה (והלכה למעשה יודע לעבוד מעל TCP\UDP) על מנת להעביר את המידע.

ולכן כאשר אומרים שמשתמשים בפרוטוקול ה-DDS בדרך כלל הכוונה היא לכמה חלקים:

- שכבת האפליקציה - כלומר האפליקציה עצמה המשתמשת ב-API שמסופק לה ע"י ה-DDS.
- שכבת ה-**DDS** - שכבה זו היא כל מה שתואר בפרק הקודם. שכבה זו מאפשר בעצם שימוש ב-DDS Domains כך שה-DomainParticipant יוכל לשלוח הודעות Public/Subscribe ל-Topic בדומיין שלו.

- שכבת ה-**RTPS** - שכבת האבסטרקציה שבין DDS לשכבת התעבורה המאפשרת באמצעות התממשקות של מערכות עם מימושים שונים של DDS לעבוד ולתקשר אחת עם השנייה.

כל זה ניתן לראות בתמונה הבאה המציגה איך הכל משתלב:



[מקור: https://fast-dds.docs.eprosima.com]

האנלוגיה המראה את הקשר בין ה-RTPS וה-DDS בצורה הכי טובה היא הקשר בין פרוטוקול **HTTP** ו- **Rest API**. ה-DDS Middleware מספקת API משוכלל במודל Publish-Subscribe עבור האפליקציה וה-RTPS אחראי לקידוד, העברה וקבלת המידע בזמן אמת.

DDS Discovery

פרוטוקול ה-DDS מספק מנגנוני זיהוי אוטומטיים למציאת DomainParticipants שרוצים לכתוב/לקרוא. זה נעשה ב-2 מצבים:

1. PDP (קיצור של Participant Discovery Phase) - בשלב זה כל ה-DomainParticipants מנסים להכיר אחד בשני. הם עושים זאת על ידי שליחת הודעות מחזוריות ב-Multicast ופורט ידועים מראש שמציגות לכולם את ה-IP והפורט שלהם, למי הם מקשיבים, לאיזה דומיין הוא שייך וכו'.

וה-DomainParticipants יכירו אחד בשני ברגע שהם יהיו חברים באותו DDS Domain. נציין שניתן גם להגדיר את הכתובות שאליהם שולחים את ההודעות ב-Unicast.

2. EDP (קיצור של Endpoint Discovery Phase) - בשלב זה ה-Publishers וה-Subscribers מכירים אחד בשני. בעזרת המידע שעובר בשלב ה-PDP הכותבים והקוראים חולקים את המידע דרך ערוץ תקשורת שהם הקימו בשלב הזה. הם שולחים אחת לשני מידע כמו ה-Topic וסוג המידע שהם כותבים/קוראים. ברגע שה-Topic וסוג המידע זהה בין 2 משתתפים הם יכולים לשלוח ולקורא מידע אחד מהשני.

שני השלבים הללו מתארים תהליך הזדהות רגיל ב-DDS דומיין אך תחת הגדרות שונות ניתן לשנות אותו מעט. לדוגמה אם נגדיר סטטית את כל ה-Publishers/Subscribers נוכל לדלג על השלב השני או נוכל להקים שרת מרכזי שאחראי לתהליך זה.

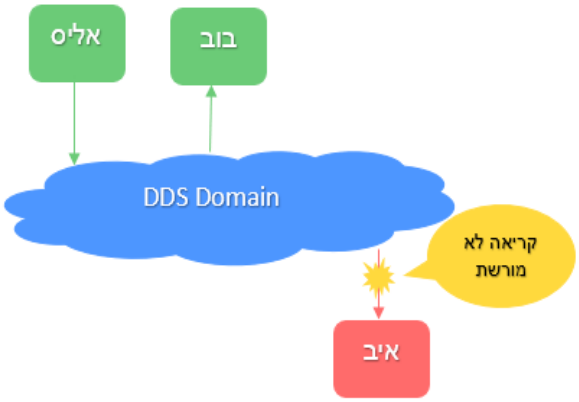
אמנם עוד לא הסברנו את תהליך ההזדהות של הישויות ב-DDS Domain, בעיקר כי זה לא ברירת מחדל בפרוטוקול, אך במידה והוא קיים זה מתרחש בשלב זה.

איומים

זה לא יהיה מאמר ב-DigitalWhisper אם לא נאמר מילה או שתיים על האיומים מולם אנחנו ניצבים ולכן לפני שנצלול להגנה שאותה Secure DDS מספק, חשוב להבין מול אילו איומים אנחנו עומדים. באופן אירוני, המעלות הכי גדולות של הפרוטוקול מהוות גם את החולשות העיקריות בו. האיומים אליו ה-DDS חשוף נובעים ממודל ה-Publish-Subscribe בו הוא תומך והודעות ה-Multicast שנשלחות ב-DDS Domain בין האפליקציות. בכללי ניתן לחלק את תרחישי האיום ל-5 סוגים:

קריאה לא מורשת

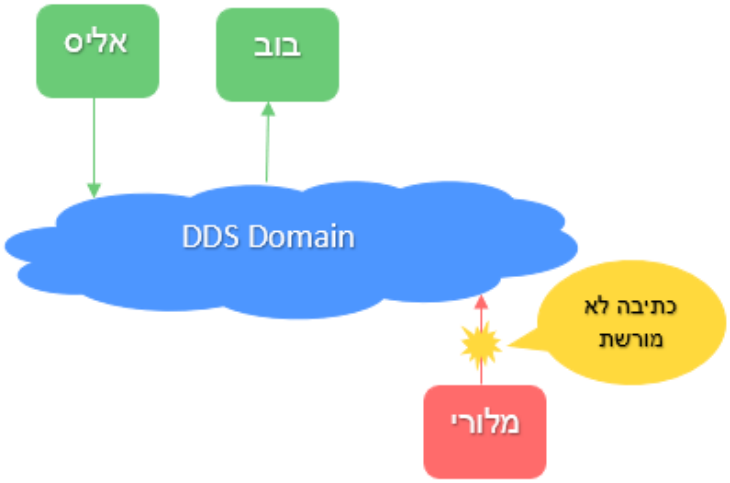
נניח שקיימים ב-DDS Domain שני DomainParticipants לגיטימיים - אליס שהיא Publisher לגיטימי ל-Topic ובוב שהוא Subscriber לגיטימי לאותו ה-Topic. בנוסף אליהם קיימת ישות שלישית בשם איב שמאזינה למידע העובר ברשת במטרה לתפוס הודעות שנשלחות מאליס לבוב. איב תוכל לעשות זאת ע"י התחברות למתג שברשת והסנפת המידע שעובר בו:



בנוסף לכך, במקרה בו אליס ובוב מתקשרים ביניהם בהודעות ב-Multicast, איב תוכל להירשם ל-Topic ולהפוך ל-Subscriber לגיטימי כחלק ממודל ה-Publish-Subscribe של הפרוטוקול.

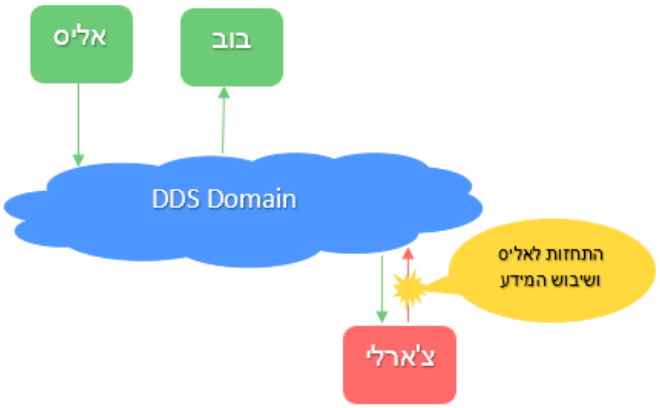
כתיבה לא מורשת

בנוסף אליהם, ישנה ישות נוספת ברשת בשם מלורי, שמעוניין להזריק לרשת הודעות זדוניות. מלורי יכול להירשם כ-Publisher ל-Topic, לשלוח הודעות ברשת כישות לגיטימית ולהזריק מידע זדוני לבוב או לכל ה-Subscribers הלגיטימיים ב-DDS Domain.



שיבוש התקשורת

איום נוסף הינו שיבוש התקשורת ברשת ע"י Subscriber לגיטימי שמתחזה ל-Publisher מאומת. נניח שבנוסף לאליס ובוב קיים DomainParticipant נוסף בשם צ'ארלי שרשום גם הוא כ-Subscribers אל אותו ה-Topic לאחר שצ'ארלי יקבל הודעה מאליס, יהיה בבעלותו ה-Headers הרלוונטיים והמזהים של אליס שיאפשרו לו להתחזות אליה ולשלוח בשמה הודעה זדונית לבוב שתראה לו לגיטימית לחלוטין.



מתקפות DoS

איום נוסף לו ה-DDS Domain חשוף הינו כל נושא מתקפות ה-DoS על כלל ה-DomainParticipants. תוקף עם גישה לרשת יכול להציף אותה בהודעות Multicast על חלק ממרחב כתובות ה-Multicast של Topic מסוים ולגרום למערכות והאפליקציות לקרוס או לאבד יכולת לספק שירותים מסוימים.

נקודות תורפה תשתיתיות

הרשאות מתירניות הן בעיה אבטחתית שלא פוסחת על אף מערכת מורכבת ואכן בארכיטקטורת DDS שלמה עלולים להיות שרתים עם פונקציונליות מסוימת שצריכים להיות מסוגלים לקבל את המידע, לאמת אותו ולהעביר אותו ליעדם, לדוגמה שרתי Proxy שונים. אמנם במקרה ושירותים אלו יקבלו הרשאות גבוהות מדי, כגון לקרוא את המידע, הם יהוו איום.

בעיה נפוצה אף יותר מבעיית הרשאות מתירניות היא misconfiguration או קינפוג חלקי. הגדרה וקינפוג לא נכון של מערכים בפרוטוקול יציבו גם את התשתית הכי בטוחה בעולם בסכנה גדולה. הדוגמה הכי טובה לכך היא מנגנוני אימות שיש בהם אופציה למשתמש ללא סיסמה.

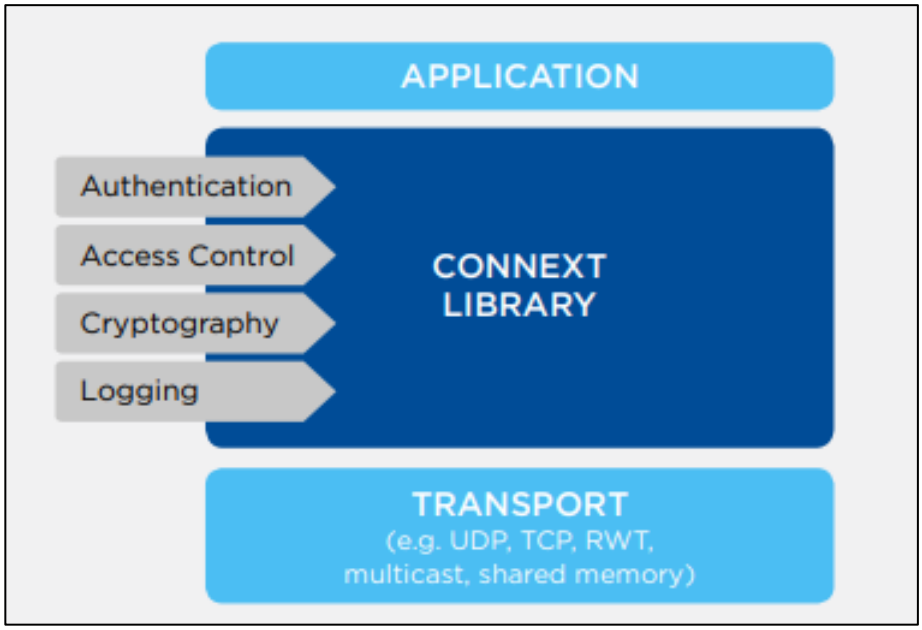
הגדרה של רכיבי הגנה בפרוטוקול DDS בהחלט עלולה להוביל למימוש האיומים שהצגנו עד כה. לדוגמה אם ניקח את איום 3 ונממש את מנגנון החתימה דרך HMAC שלכאורה אמור למנוע מתוקף לשלוח הודעות משלו כי אינו יכול לחתום אותם בעצמו. כדי לממש זאת אליס ובוב יגדירו בינם מפתח סודי וישתמשו בו עבור ה-HMAC. אך גם בתרחיש זה אנו חשופים לכמה בעיות:

- צ'ארלי יכול לקחת את ההודעות של אליס ולשלוח אותם כשלו. גם להשתמש ב-HMAC של אליס (replay attack) וגם לקחת את כל ה-headers הרלוונטיים של אליס עבור ההודעות שלו.
- במקרה בו הגדירו <u>מפתח סודי אחד לכל המשתמשים</u> (לצערי זה קורה יותר ממה שאתם חושבים) אז המפתח הסודי יהיה בבעלותו של צ'ארלי והוא יוכל לחתום עם ה-HMAC בעצמו.

כמובן קיימים פתרונות פשוטים שמונעים את האיומים הללו, כגון תהליך הזדהות מלא כפי שיפורט בפרקים הבאים או להשתמש במפתח ייחודי בין כל זוג ישויות. אך גם כאן חשוב להביא הגדרות ברורות וחד משמעיות כדי להימנע מניצולם בדרכי Downgrade Attacks או כאשר מאפשרים כמה דרכים לחתום אז התוקף יבחר את הפתרון הפחות בטוח. ככלל חשוב להבין כי שום מנגנון לא יוכל להגן מפני הגדרה לא נכונה של מערכת.

Security Plugins

אז איך בעצם ניתן להגן על DDS ולאפשר פעילות ב-Secure DDS? ובכן, כדי להגן על ה-DDS (ובעצם להפוך אותו ל-Secure DDS) קיימים מספר Plugins המספקים הגנה על סודיות, אמינות וזמינות המידע ב- DDS Domain. יכולות ההגנה אותן ה-DDS מספק כוללות יכולת להגבלת גישה למידע, יכולת לאימות ה-DomainParticipants, יכולת להצפנת המידע שנשלח בין ה-Publishers ל-Subscribers ויכולת לשמירת לוגים מפורטים. בנוסף לכך, ניתן להיעזר בהגדרות QoS כדי לתעדף מידע קריטי ברשת, להגדיר קצבים להעברת המידע ובכך להגן על הרשת מהתקפות DoS.



[מקור: https://info.rti.com/hubfs/Datasheets/RTI_Datasheet_10018_Connext-DDS-Secure_V7_Web_0718.pdf]

אז הגיע הזמן שנדבר יותר לעומק על הגנה, לא?

כעת, נסביר לעומק על ה-Plugins העיקריים שיש ל-Secure DDS להציע ותוך כדי נראה איך הם מגנים אל מול האיומים שפרטנו בפרק הקודם.

Authentication Plugin

כאשר משתמשים ב-**Authentication Plugin**, כל DomainParticipant שמעוניין להצטרף ל-DDS Domain נדרש לעבור תהליך של אימות. בעצם כל DomainParticipant שמשתמש ב- Pluginהזה, יוכל לתקשר אך ורק עם DomainParticipants אחרים שגם כן משתמשים בו. חשוב להדגיש שבלי ה-Plugin הזה, אין אכיפה על האפליקציות והמערכות שמצטרפות ל-DDS Domain. ולכן ה-Authentication Plugin מספק הגנה מול איומים מספר 1 ו-2 - מי שלא יכול להתאמת מול ה-DDS Domain בין היתר לא יכול לכתוב או לקרוא Topics הנמצאים בו.

ה-Authentication Plugin תומך ב-2 שיטות הזדהות:

1. הזדהות מבוססת תעודות X.509 - זו היא השיטה הנפוצה ביותר כאשר משתמשים ב-**DDS**, בשיטה זו כל DomainParticipant שומר מספר פרמטרים:

- **Identity CA Certificate** - תעודה מסוג X.509 המכילה את המפתח הציבורי של ה-Identity CA, החותם על כלל התעודות של ה-DomainParticipants.
- **Identity Certificate** - תעודה מסוג X.509 המכילה את המפתח הציבורי של ה-DomainParticipant וחתומה ע"י המפתח הפרטי של ה-Identity CA.
- **Private Key** - המפתח הפרטי של ה-DomainParticipant.

שני DomainParticipants מבצעים אימות הדדי בעזרת Handshake תוך שימוש באלגוריתם RSA או ECDSA. לאחר תהליך האימות שני הצדדים יסנכרנו ביניהם מפתח סימטרי להצפנת התווך ביניהם תוך שימוש באלגוריתם DH או ECDH (במקרה בו יהיה שימוש גם ב-Cryptography Plugin עליו נרחיב בהמשך).

הערה: לא נרחיב לעומק על תהליך אימות התעודות והודעות ה-Handshake שעוברות בין שני DomainParticipants כיוון שזה לא תהליך ייחודי ל-DDS אלא תהליך ידוע ונפוץ שמכיל שימוש באלגוריתמים נפוצים בתחום זה. אם מישהו מעוניין לצלול לעומק של עולם זה, קיים מאמר מצוין ב- Digital Whisper שנכתב ע"י יהונתן אלקבס שמסביר את כל הנושא לעומק: על תעודות דיגיטליות - איך לסמוך על הסינים

ניתן לראות בדוגמא הבאה קובץ XML בו מוגדר תהליך אימות ב-DDS Domain על בסיס תעודות דיגיטליות ואלגוריתם סנכרון מפתחות DH, עבור כלל ה-DomainParticipants:

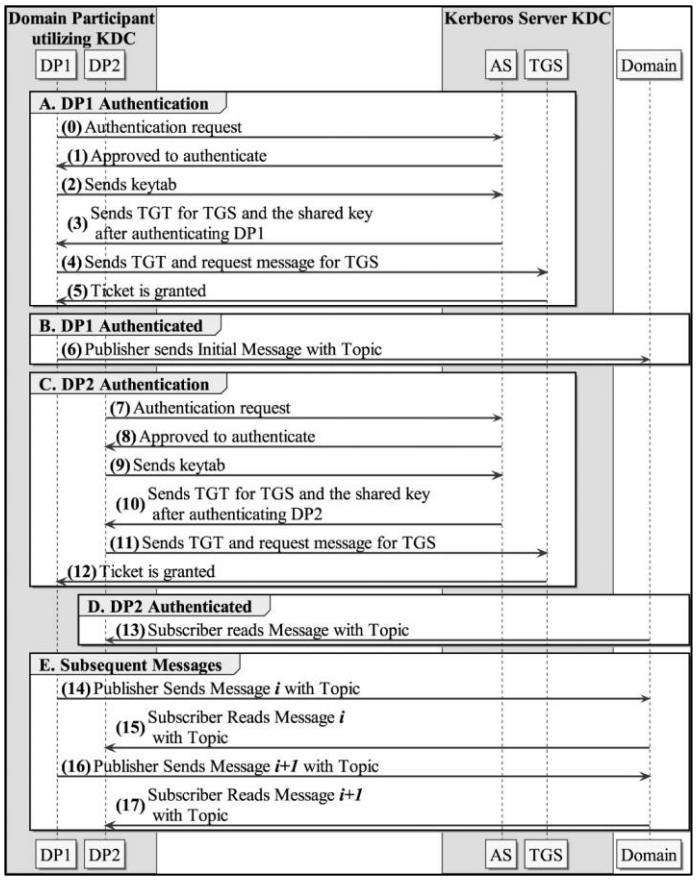
```
<participant profile_name="secure_domainparticipant_conf_auth_plugin_xml_profile">
    <rtps>
        <propertiesPolicy>
            <properties>
                <!-- Activate DDS:Auth:PKI-DH plugin -->
                <property>
                    <name>dds.sec.auth.plugin</name>
                    <value>builtin.PKI-DH</value>
                </property>
                <!-- Configure DDS:Auth:PKI-DH plugin -->
                <property>
                    <name>dds.sec.auth.builtin.PKI-DH.identity_ca</name>
                    <value>/path/to/CAcert/identity_ca.pem</value>
                </property>
                <property>
                    <name>dds.sec.auth.builtin.PKI-DH.identity_certificate</name>
                    <value>/path/to/identity/cert/identity.pem</value>
                </property>
                <property>
                    <name>dds.sec.auth.builtin.PKI-DH.private_key</name>
                    <value>/path/to/private/private.pem</value>
                </property>
            </properties>
        </propertiesPolicy>
    </rtps>
</participant>
```

2. **DDS-C**: מדובר ב-Plugin העונה על השאלה מה קורה אם נשלב בין DDS ו-Kerberos Authentication - והתשובה הקצרה נקבל **DDS-Cerberus**. על Kerberos לא נרחיב, אבל למעוניינים ניתן לקרוא את המאמר של נתנאל כהן (אחד מכותבי המאמר הנוכחי) ועדי מליאנקר:

1-st Step to Tame a Kerberos: Know Your Enemy

כדי להבין איך DDS-C עובד עלינו להכיר את הקובץ **Keytab**, הוא מכיל את המפתח הסודי (לא ה-tickets אלא מפתח משתמש, מפתח השיחה וכו') עבור SPN-ים (למשל HTTP/srv.exmpl.com@EXAMPLE.COM) באימות Kerberos. קובץ זה שמור בצורה המאפשרת להשתמש בו כדי לפענח את ההודעות בלי לגלות את המפתח הסודי עצמו. קובץ זה קיים לכל DomainParticipant.

אוקיי, נשמע מגניב אבל איך תכל'ס קורא תהליך האימות עצמו? ובכן, לשם כך נסתכל על הדיאגרמה הבאה:



[מקור: https://link.springer.com]

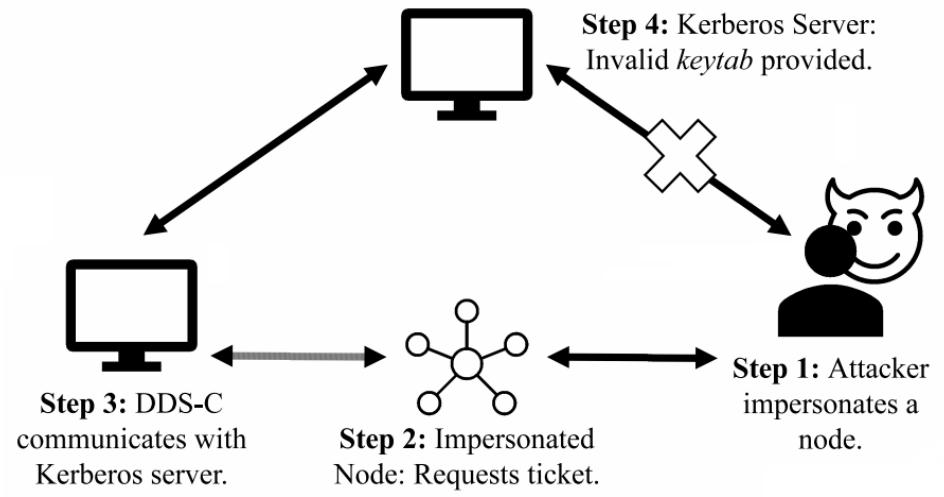
כפי שניתן לראות תהליך ה-Discovery דורש שרת מרכזי - KDC כלומר כל תהליך ההזדהות יתבצע דרכו או הישות לא תהיה חלק מה-DDS Domain. בואו נפשט את השלבים:

1. ראשית נאמת את עצמנו בעזרת קובץ ה-Keytab מול ה-KDC.
2. אם האימות עבר בהצלחה נקבל את ה-TGT שזה יהיה ה-SSO שלנו. כלומר אם אנו חברים בכמה DDS Domains נשתמש באותו TGT כדי לקבל את ה-TGS עבור כל דומיין (כמובן בהתאם להרשאות של הישות).

3. לאחר מכן נבקש מרכיב ה-TGS להיות חלק מדומיין ספציפי ונקבל את ה-TGS עבורו.

תהליך ההזדהות של כל DomainParticipant נעשה אך ורק כאשר הוא מצטרף ל-DDS Domain. כל בקשות הקריאה והכתיבה שיתבצעו אחרי יתבצעו כרגיל כפי שתואר בפרקים הקודמים. ניתן לראות זאת בשלבים B, D ו-E.

נדגיש שתוכן ההודעות זהה לתהליך ההזדהות הרגיל של פרוטוקול ה-Kerberos ולכן עם טיפה שינויים לתצורה החדשה יהיה ניתן להשתמש ברוב המתקפות המוכרות. אבל במצב הרגיל איך זה מגן על התקשורת מהתוקף?



[מקור: https://apps.dtic.mil/sti/trecms/pdf/AD1166912.pdf]

ניתן לראות שעקב ה-Plugin של ההזדהות התוקף לא יכול לכתוב/לקרוא מהמשאבים ב-DDS Domain. ובלי קובץ ה-Keytab עם סודות הישות הוא גם לא יצליח להתאמת, כלומר התוקף אינו יכול להתחזות למשתמש בדומיין אלא אם הוא גנב את קובץ ה-**Keytab** בעצמו.

XML לדוגמא (נכתב בפורמט גנרי לשימוש בהגדרות Security של DDS):

```
<dds>
    <!-- Configure Security Plugin for Authentication -->
    <security>
        <authentication_plugin>
            <type>Kerberos</type>
            <config>
                    <!-- Specify Kerberos Service Principal -->
                <kerberos_service_principal>dds-service@YOUR-REALM.COM
                </kerberos_service_principal>

                <!-- Path to Keytab File for Service Principal -->
                <kerberos_keytab>/path/to/dds-service.keytab</kerberos_keytab>

                <!-- Specify Ticket Renewal and Lifetime Settings -->
                <renewable_tickets>true</renewable_tickets>
                <ticket_lifetime>
                    <seconds>3600</seconds>
                </ticket_lifetime>
            </config>
        </authentication_plugin>
  </security>
</dds>
```

Access Control Plugin

כפי שניתן להבין מהשם, Plugin זה מספק יכולת להגביל ולאמת את ההרשאות של כל DomainParticipant לאחר סיום ההזדהות שלו ב-Domain. ניתן להגדיר לאיזה DomainPatricipant יש הרשאות של יצירה, הריסה, קריאה וכתיבה לאיזה Topic. לשם כך, כל DomainParticipant ישמור את הפרמטרים הבאים:

1. **Permissions CA Certificate** - תעודה מסוג X.509 המכילה את המפתח הציבורי של ה-Permissions CA, החותם על מסמכי ההרשאות של כל ה-DomainParticipants.
2. **Governance** - מסמך XML המפרט כיצד נדרש לאבטח את ה-DDS Domain ואת כלל ה-Topics בו, החתום ע"י ה-Permissions CA.
3. **Permission** - מסמך XML המפרט אילו הרשאות יש ל-DomainParticipant, החתום ע"י ה-Permissions CA.

מה זה בדיוק אומר להגן ה-DDS Domain או על ה-Topics? אילו הרשאות בדיוק אפשר להגדיר לכל DomainParticipants? בשביל זה אנחנו צריכים לצלול לתוך קבצי ה-Governance וה-Permission ולהבין אילו יכולות והגדרות הם מספקים לנו כדי להגן על המידע שלנו.

מסמך ה-Governance מכיל מספר חוקים אפשריים שניתן להגדיר עבור ה-DDS Domain ועבור ה-Topics:

Domain Rules		
Allow Unauthenticated Participants	קובע האם DomainParticipants מחויבים לבצע הזדהות בעת כניסתם ל-Domain והאם הם רשאים לתקשר עם Participants שלא ביצעו הזדהות מוצלחת.	**False** - חובה לבצע הזדהות. **True** - אין חובה לבצע הזדהות, ניתן לתקשר עם Participant שלא ביצע הזדהות.
Enable Join Access Control	קובע האם נדרש לאכוף את ה-Permissions של ה-Domain Participants.	**False** - ה-Permissions לא יבדקו. **True** - ה-Permissions יבדקו.
Discovery Protection Kind	קובע האם תהליך הזיהוי בו Domain Participants מגלים אחד את השני ואת ה-Topics צריך להיות מוגן וכיצד.	**NONE** - לא נדרשת הגנה. **SIGN** - ההודעות בתהליך הזיהוי צריכות להיות חתומות. **ENCRYPT** - ההודעות בתהליך הזיהוי צריכות להיות מוצפנות.
Liveliness Protection Kind	קובע האם הודעות Liveliness יהיו מוגנות וכיצד (הודעות ש-Domain Participants שולחים זה לזה כדי לבדוק את זמינותם).	**NONE** - לא נדרשת הגנה. **SIGN** - ההודעות צריכות להיות חתומות. **ENCRYPT** - ההודעות צריכות להיות מוצפנות.
RTPS Protection Kind	קובע האם הודעות RTPS יהיו מוגנות וכיצד (Real-Time Publish-Subscribe הוא הפרוטוקול בו ה- Domain Participants שולחים הודעות זה לזה).	**NONE** - לא נדרשת הגנה. **SIGN** - ההודעות צריכות להיות חתומות. **ENCRYPT** - ההודעות צריכות להיות מוצפנות.

Topic Rules		
Enable Discovery Protection	קובע האם להחיל את ההגנה שנקבעה בחוק "Discovery Protection Kind" על ה-Topic הספציפי הזה.	**False** - לא נדרשת הגנה. **True** - ההודעות ישלחו בצורה מאובטחת כפי שנקבע ב-Domain Rule.
Enable Liveliness Protection	קובע האם להחיל את ההגנה שנקבעה בחוק "Liveliness Protection Kind" על ה-Topic הספציפי הזה.	**False** - לא נדרשת הגנה. **True** - ההודעות ישלחו בצורה מאובטחת כפי שנקבע ב-Domain Rule.
Enable Read Access Control	קובע האם יש לאכוף הרשאות קריאה (Subscribe) ל-Topic הספציפי הזה.	**False** - לא נדרש לאכוף הרשאות. **True** - נדרש לאכוף הרשאות.
Enable Write Access Control	קובע האם יש לאכוף הרשאות כתיבה (Publish) ל-Topic הספציפי הזה.	**False** - לא נדרש לאכוף הרשאות. **True** - נדרש לאכוף הרשאות.
Metadata Protection Kind	קובע האם וכיצד יש להגן על ה-Metadata של הודעות RTPS הקשורות ל-Topic הספציפי הזה.	**NONE** - לא נדרשת הגנה. **SIGN** - נדרשת חתימה. **ENCRYPT** - נדרשת הצפנה.
Data Protection Kind	קובע האם וכיצד יש להגן על ה-Payload של הודעות RTPS הקשורות ל-Topic הספציפי הזה.	**NONE** - לא נדרשת הגנה. **SIGN** - נדרשת חתימה. **ENCRYPT** - נדרשת הצפנה.

בדוגמא הבאה ניתן לראות מימוש של קובץ Governance עבור מימוש Fast DDS. הקובץ מתייחס ל- DDS Domain בעל ID עם ערך 10 המכיל שני Topics: foo ו-bar. לכל Topic יש חוקים משלו כפי שהוצגה בטבלה עם ערכים שונים. בנוסף לכך ניתן לראות בקובץ את ה-Domain Rules אותם סקרנו:

```
<dds>
    <domain_access_rules>
        <domain_rule>
            <!-- Domain ID -->
            <domains>
                <id>10</id>
            </domains>

            <!-- Domain Rules -->
            <allow_unauthenticated_participants>false
            </allow_unauthenticated_participants>
            <enable_join_access_control>true</enable_join_access_control>
            <discovery_protection_kind>ENCRYPT</discovery_protection_kind>
            <liveliness_protection_kind>ENCRYPT</liveliness_protection_kind>
            <rtps_protection_kind>ENCRYPT</rtps_protection_kind>

            <topic_access_rules>
                <!-- Topic foo Rules -->
                <topic_rule>
                    <topic_expression>foo</topic_expression>
                    <enable_discovery_protection>true
                    </enable_discovery_protection>
                    <enable_liveliness_protection>false
                    </enable_liveliness_protection>
                    <enable_read_access_control>true</enable_read_access_control>
                    <enable_write_access_control>true
                    </enable_write_access_control>
                    <metadata_protection_kind>ENCRYPT</metadata_protection_kind>
                    <data_protection_kind>ENCRYPT</data_protection_kind>
                </topic_rule>

                <!-- Topic bar Rules -->
                <topic_rule>
                    <topic_expression>bar</topic_expression>
             <enable_discovery_protection>false
             </enable_discovery_protection>
                    <enable_liveliness_protection>false
                    </enable_liveliness_protection>
                    <enable_read_access_control>true</enable_read_access_control>
                    <enable_write_access_control>true
                    </enable_write_access_control>
                    <metadata_protection_kind>SIGN</metadata_protection_kind>
                    <data_protection_kind>ENCRYPT</data_protection_kind>
                </topic_rule>
            </topic_access_rules>
        </domain_rule>
    </domain_access_rules>
</dds>
```

קובץ ה-Permissions מכיל סט של הרשאות עבור כל DomainParticipant, המפרט האם לו גישה של קריאה או כתיבה ל-Topics מסוימים ב-Domains מסוימים. ניתן להגדיר את ההרשאות של המשתמשים בתצורה Whitelist בעזרת Allow Rules או בתצורה Blacklist בעזרת Deny Rules. ניתן אף להגביל את ההרשאות שניתנות בקובץ זה לטווח של תאריכים אליהם הוא יהיה רלוונטי.

בדוגמא הבאה ניתן לראות מימוש של קובץ Permissions עבור מימוש Fast DDS. הקובץ מגדיר הרשאות לשני DomainParticipants בשם P1 ו-P2 עבור Topic בשם foo בתוך DDS Domain בעל ID עם ערך 10. עבור P1 יוגדרו הרשאות כתיבה ל-Topic ועבור P2 יוגדרו הרשאות קריאה בעזרת Allow Rules, כאשר ההרשאות של שתי המשתמשים יהיו תחומות בטווח זמנים מסוים:

```
<dds>
    <permissions>
        <!-- P1 Permissions Set -->
        <grant name="P1Permissions">
            <subject_name>emailAddress=P1@gmail.com</subject_name>
            <validity>
                <not_before>2020-01-01T12:00:00</not_before>
                <not_after>2029-12-31T12:00:00</not_after>
            </validity>
            <allow_rule>
                <domains>
                    <id>10</id>
                </domains>
                <!-- Permit P1 to publish in Topic foo -->
                <publish>
                    <topics>
                        <topic>foo</topic>
                    </topics>
                </publish>
            </allow_rule>
            <default>DENY</default>
        </grant>
        <!-- P2 Permissions Set -->
        <grant name="P2Permissions">
            <subject_name> emailAddress=P2@gmail.com</subject_name>
            <validity>
                <not_before>2020-01-01T12:00:00</not_before>
                <not_after>2029-12-31T12:00:00</not_after>
            </validity>
            <allow_rule>
                <domains>
                     <id>10</id>
                </domains>
                <!-- Permit P2 to subscribe in Topic foo -->
                <subscribe>
                    <topics>
                        <topic>foo</topic>
                    </topics>
                </subscribe>
            </allow_rule>
            <default>DENY</default>
        </grant>
    </permissions>
</dds>
```

אם לסכם את ה-Access Control Plugin - מדובר ביכולת ההגנה הכי מורכבת למימוש ב-Secure DDS. נדרש להבין לעומק אילו משתמשים ניגשים לאילו Topics ומעבר לכך, אילו הרשאות בדיוק כל אחד צריך מהם. עם זאת, מדובר ביכולת שמספקת הגנה בצורה הדוקה ביותר כאשר היא מוגדרת כשורה. אם נחזור

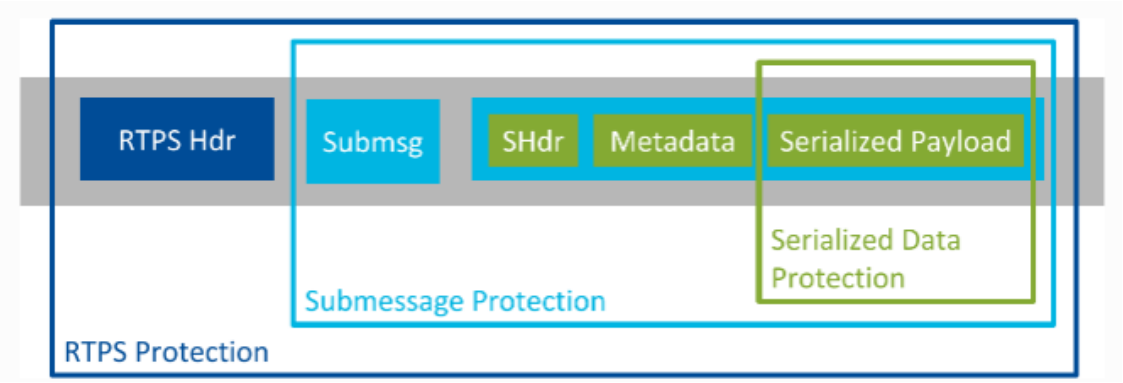
לניתוח האיומים שבוצע קודם לכן, ע"י הגדרת קובץ Governance ניתן להגן על ה-Domain וההודעות הנשלחות בו אל מול איום מספר 3. ע"י הגדרת קובץ Permissions ניתן להגן על כלל ה-Topics מגישה לא מורשת אל מול איומים מספר 1 ו-2.

Cryptographic Plugin

מספק יכולות הגנה על סודיות ו/או אמינות בשכבות שונות עבור ההודעות הנשלחות ב-DDS Domain. ה-Plugin הזה מתקשר לשני ה-Plugins הקודמים שסקרנו: תחת Authentication ניתן לקבוע איזה גודל מפתח ואלגוריתם הצפנה ה-DomainParticipants יסנכרנו לאחר תהליך הזדהות מוצלח. תחת Access Control ניתן לקבוע באיזה אלגוריתם ייחתם או יוצפן מידע כמו הודעות RTPS, Discovery ו-Liveliness.

ה-Plugin מספק גמישות בנוגע לשכבות השונות של המידע עליהן הוא מספק הגנה:

- ניתן להגן על כל הודעת ה-RTPS שעוברת ב-DDS Domain.
- ניתן להגן על ה-Submessage המכיל את ה-Metadata, Payload ומספר Headers נוספים.
- ניתן להגן על ה-Payload בלבד.



[מקור: https://community.rti.com]

ההגנה על המידע הזה מתבצעת בעזרת לרוב ע"י אלגוריתם AES-GCM המספק הצפנה על המידע והוספת שדה MAC כדי לוודא את אמינותה. ה-Plugin תומך בגודל מפתח AES של 128-bit או 256-bit. המפתח הזה נדרש לא רק לשם ההצפנה, אלא גם כדי ליצור את ערך ה-MAC להודעות. במקרים בהם נדרש אך ורק לחתום ע"י המידע ללא הצפנתו, ניתן להשתמש ב-AES-GCM כדי ליצור את שדה ה-MAC או שניתן להשתמש באלגוריתם HMAC-256 כאלטרנטיבה.

מבחינת איומים, יכולת של הצפנה ושמירה על אמינות המידע מספקת לנו הגנה ברורה אל מול איום 1- קריאה לא מורשת, כאשר גורם זדוני מנסה להאזין לרשת דרך המתג (Eavesdropping) ומול איום 2- כתיבה לא מורשת, לתרחיש בו גורם זדוני מנסה לשנות את המידע בהתקפת Man-in-the-Middle.

Logging Plugin

מספק יכולת תיעוד לאירועי אבטחה שנוצרים ע"י שאר ה-Plugins שסקרנו במאמר. ניתן לתעד לוגים של התנהגות צפויה או של בעיות אבטחה בצורה רחבה יותר מאשר הלוגים אותם ניתן להגדיר דרך הגדרות ה-QoS.

מבנה הלוגים מכיל את השדות הבאים:

שדה	הגדרה
Timestamp	התאריך והשעה בה האירוע תועד
Category	הקטגוריה אליה הלוג שייך
Verbosity Level	רמת החומרה של הלוג
Message	ההודעה אותה הלוג מציג
File Context	הקובץ והשורה שמהווים מקור לאירוע שתועד
Function Name	הפונקציה שמהווים מקור לאירוע שתועד

כמו כן, ה-Plugin מאפשר להגדיר מספר רחב יותר של רמות חומרה ללוגים אותם הוא מתעד כגון:

רמת חומרה	הגדרה
EMERGENCY_LEVEL	המערכת מתנהגת בצורה לא רגילה, כדאי לעצור את פעילותה
ALERT_LEVEL	ישנה בעיה שכדאי לתקן במיידיות
CRITICAL_LEVEL	יש כישלון או בעיה באפליקציה מרכזית
ERROR_LEVEL	נמצאה בעיה כללית
WARNING_LEVEL	עלול להצביע על בעיה עתידית אם לא תתבצע פעולה
NOTICE_LEVEL	פעילות חריגה שלא בהכרח מצביעה על בעיה
INFORMATIONAL_LEVEL	פעולה תקינה, אין צורך לבצע פעולה
DEBUG_LEVEL	תיעוד פעולה תקינה לשם בדיקות

QoS Settings

הגדרות QoS ב-DDS אינן חלק מפורש מ-Secure DDS Plugins. אם זאת, הן הרוויחו מקום של כבוד במאמר בשל מספר הגדרות יכולות לספק הגנה נוספת עבור ה-DDS Domain, כנגד מספר התקפות ששאר ה-Plugins לא נותנים להם מענה:

- **ReliabilityQosPolicy** - הפוליסה מוודא כי ההודעות אכן הגיעו אל ה-Subscriber. במקרה וההודעות נפלו בדרך הן ישלחו בשנית. מספק הגנה עבור אמינות והזמינות המידע עבור גורם זדוני שמנסה לשבש את התעבורה ב-DDS Domain.
- **TimeBasedFilterQosPolicy** - הפוליסה מגדירה ערך של חלון זמנים בו היא מצפה לקבל לא יותר ב-Sample אחד, מספק הגנה מול התקפות DoS למיניהן. הפוליסה הזו מתקשרת ישירות לאיום מספר 4 שפרטנו בפרק הקודם.

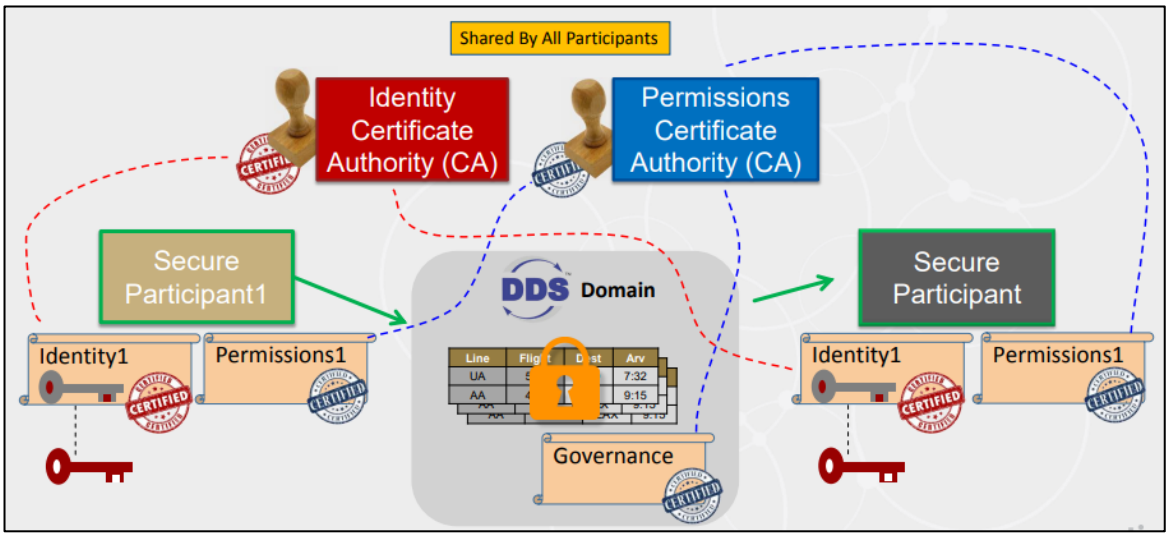
- **ResourceLimitsQosPolicy** - הפוליסה מגדירה את כמות המשאבים הזמינים עבור האפליקציה המשתמשת ב-DDS, מספק הגנה מול DomainParticipants זדוניים שמנסים לנצל משאבים ברשת.

- **DeadlineQosPolicy** - הפוליסה מגדירה חלון זמנים מקסימלי בו אפליקציה אמורה לשלוח מידע. מספק הגנה מול DomainParticipants זדוניים שמנסים לפגוע ביכולות ה-Real-Time של המערכת ובך לפגוע באמינותה.

סיכום

פרוטוקול ה-DDS הוא פרוטוקול מתקדם המשמש לתקשורת במערכות מבוססות זמן אמת. במאמר זה, סקרנו את היכולות והשימושים של הפרוטוקול, את האיומים הפוטנציאליים שאליו הוא חשוף, והתעמקנו בהגנות המרכזיות הנדרשות כדי לשמור על פרטיות ובטיחות המידע והמשתמשים.

כל יכולת הגנה שפרטנו נותנת מענה חלקי לאיומים הללו אך חיבור של כולם יחדיו מספק מעטפת הגנה שלמה יותר עבור DDS Domain שיכולה להקשות מאוד על תוקף לפגוע במערכת או בנכס שמבצע שימוש בפרוטוקול. אם נחבר את כל חתיכות הפאזל שתיארנו במהלך המאמר, נקבל Secure DDS Domain המאופיין בצורה הבאה:



[מקור: https://ruffsl.github.io/IROS2018_SROS2_Tutorial/content/slides/Background.pdf]

ה-Domain יכיל שני ישויות Identity CA - CA ו-Permissions CA. כל DomainParticipant יכיל תעודה דיגיטלית לשם הזדהות ב-Domain החתומה ע"י ה-Identity CA.

בנוסף לכך, הוא יכיל קובץ Permissions המכיל אילו הרשאות כתיבה או קריאה קיימות לו ולאילו Topics, כאשר קובץ זה חתום ע"י ה-Permissions CA. ה-Domain עצמו מוגן ע"י קובץ Governance שקובע האם לאכוף את הזדהות או הרשאות ב-Domain ואיזה חלק מתוך המידע שנשלח בו יהיה מוצפן.

על הכותבים

יותם דוד, בן 27, ארכיטקט סייבר.

נתנאל כהן, בן 25, חוקר אבטחת מידע ומתקפות דומיין.

לשאלות וכל דבר אחר ניתן לפנות אלינו ישירות. תודה שקראתם!