Digital Whisper Technologic Papers

על שיניים כחולות ופרצות זדוניות: פרוטוקול Bluetooth על קצה המזלג

מאת איתמר שבתאי עמיעזר ודניאל שוסטק

היסטוריה

השנה היא 1989. חברת אריקסון מובייל (Ericsson Mobile), יצרנית מוצרי תקשורת מחליטה להתחיל לפתח טכנולוגיה חדשה, טכנולוגיית רדיו "short-link", בשאיפה לייצר אוזניות אלחוטיות.

עד אז, מרבית תהליכי החיבור בין מכשירים השתמשו בכבלים (למשל, כבלי RS-232[1]) אך אלו היו מסורבלים, מבולגנים וכמובן דרשו חלל פיזי. פתרון נוסף שהיה מקובל הוא שימוש בקרני אינפרא רד (Infrared). הפתרון הזה משומש עד היום בשלטים בשימוש בעדשה שמרכזת את הקרינה אל עבר המטרה ובכך מפעילה אותה, אך החסרון המרכזי הוא שהיא שימושית בעיקר בחללים סגורים (הקרינה לא עוברת דרך הקירות) ודורשת עדשה וכיוון מסוים. בחללים פתוחים הקרינה עלולה להסב נזק וקשה למקדה, ובמכשירים נייחים רבים קשה עד בלתי אפשרי להשתמש בקרינה כדרך העברת תקשורת בין המכשיר למכשיר אחר.

המחסור בפתרונות קיימים והקריאה הנרחבת לטכנולוגיה גרמו לאריקסון לפתח את פרויקט "short-link" ולהחליט להתחיל לפעול בטכנולוגיה שפיתחו במטרה למצוא פתרון חדש להחלפת שיטות החיבור הקיימות.

בין השנים 1994-1997, אריקסון המשיכה לפתח את הטכנולוגיה בשיתוף עם IBM עד 1997, שם פנו יחד IBM לחברת ThinkPad בנוגע לשיתוף פעולה. בשיתוף הפעולה, הוחלט לשלב את טכנולוגיות ה"-short link" גם במחשבי הנייד של ThinkPad וגם בטלפונים של אריקסון.

בגלל נתח השוק הקטן, הוחלט להפוך את הטכנולוגיה לתקן פתוח בתעשייה במטרה להפיצה. לאחר גיוס אינטל (Intel), נוקיה (Nokia) וטושיבה (Toshiba) למחקר, במאי 1998 הוקמה קבוצת ה-Bluetooth SIG. בשנת 1999 הושקו אוזניות ה-Bluetooth הראשונות, והטלפון Ericsson T36 ששילב את הטכנולוגיה.

---

[1] פרוטוקול העברת מידע בשכבה הפיזית.

באוקטובר 2001 IBM הציגה את IBM ThinkPad A30 שהיה ללפטופ הראשון אשר השתמש בטכנולוגיה. משם, הטכנולוגיה המשיכה להתפתח והופצה ברחבי העולם עד שהגיעה למצב בו אנחנו מכירים אותה היום.

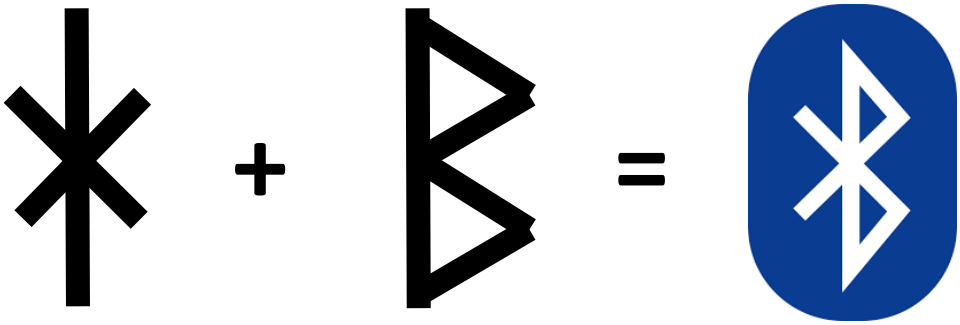
[ThinkPad A30]

[Ericsson T36]

מקור השם

השם Bluetooth היא מילה מאונגלזת מסקנדינבית שהייתה כינויו של המלך האראלד הראשון, מלך דנמרק. הוא ידוע בכך שאיחד את השבטים הדניים תחת ממלכה אחת, ולכן ב-1997 הוצע הרעיון לשם כיוון שהפרוטוקול שואף לאחד את כל פרוטוקולי התקשורת הקיימים בתקן אחיד בדומה לעשייתו של המלך.

הלוגו עצמו הוא שילוב של האותיות "הגלז" ו-"ברקנן" מאלפבית הרונות ששימשה בסקנדינביה ואלו ראשי התיבות של שמו של המלך:



רקע תיאורטי והסבר טכני

תהליך החיבור

Bluetooth הינו פרוטוקול המתבסס על יצירת קשר בין master ל-slave. הרעיון הוא שלאחר שקשר שכזה נוצר, החיבור נשמר עד לשבירה או ניתוק מכוון. יצירת חיבור ראשוני בין שני מכשירי Bluetooth שונים הוא תהליך המורכב משלושה שלבים עיקריים:

- סריקה: במידה ושני התקני Bluetooth זרים שואפים ליצור קשר ולא יודעים דבר זה על זה, חייבת להתחיל סריקה על ידי אחד מן המכשירים כדי לגלות את השני. מכשיר אחד שולח את בקשת הבירור, וכל מכשיר שמאזין לבקשה הזו יגיב עם הכתובת שלו. לאחר מכן, ניתן לבקש מידע על ה-services (שירותים) שכל מכשיר מציע.

- זימון: הזימון הוא תהליך יצירת החיבור בין שני התקני ה-Bluetooth. כעת, כל מכשיר יודע את הכתובת של השני וניתן להתחיל בתהליך החיבור.

- חיבור: לאחר שהזימון הושלם, המכשיר נכנס למצב חיבור. המכשירים מוגדרים כמקושרים.

נראה דוגמא של הודעת זימון ב-Wireshark:

```
Bluetooth L2CAP Protocol
    Length: 8
    CID: L2CAP Signaling Channel (0x0001)
  Command: Connection Request
      Command Code: Connection Request (0x02)
      Command Identifier: 0x0a
      Command Length: 4
      PSM: SDP (0x0001)
      Source CID: Dynamically Allocated Channel (0x0052)
```

הודעה זו מבקשת תחילת חיבור עם המכשיר.

מצבי חיבור

לפני שנפרט על כל אחד מתהליכי החיבור, נכיר מושג חדש: מצבי חיבור.

בעת הקשר, ישנם ארבעה מצבים אופציונליים בהם כל מכשיר יכול להיות:

- active mode: המצב השגרתי והדיפולטי של קשר, בו המכשיר נמצא במצב פעיל של קבלת מידע או שליחת מידע.
- sleep/sniff mode: המצב מוגדר כמצב חיסכון בסוללה. המכשיר מוגדר כפחות פעיל, ובמקום לקחת חלק אקטיבי הוא "יישן" ויאזין אך ורק לשידורים במרווח זמן מוגדר (לדוגמה, כל 80 אלפיות השנייה).
- hold mode: hold mode הוא שילוב בין שני המצבים הראשונים, כאשר מוגדרת עבור המכשיר תקופת זמן קבועה מראש בה יפעל ode, sleep ומיד לאחר מכן חוזר למצב פעיל. ה-master יכול לפקוד על ה-slave לעבור למצב הזה.

- connection park mode: מצב של השבתה מוחלטת. ה-master יכול לפקוד על slave לעבור למצב הזה, וה-slave יהפוך ללא פעיל בכלל עד שה-master יגיד לו להתעורר בחזרה.

שכבות הפרוטוקול

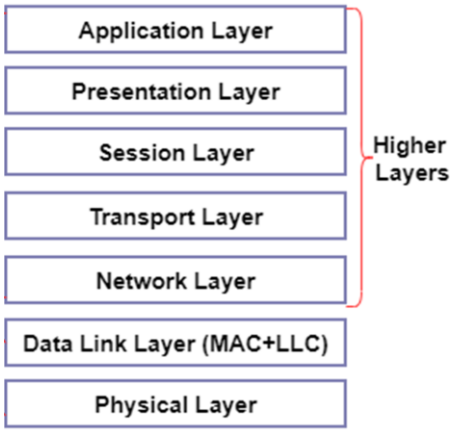
לפרוטוקול Bluetooth ישנו מודל המורכב משלוש שכבות עיקריות, ומסביר את תהליך ההתקשרות:

- **שכבה ראשונה: השכבה הפיזית** - השכבה הפיזית בפרוטוקול Bluetooth אחראית לשידור וקליטת נתונים. היא מגדירה את מפרטי החומרה (תדרי רדיו ומאפיינים פיזיים רלוונטים אחרים). השכבה הפיזית משתמשת בפס 2.4 ISM[3] GHz[2] ומשתמש בטכניקות כמו קפיצת תדר מרווח (FHSS[4]) כדי למזער הפרעות. השכבה הפיזית מטפלת בשידור של נתונים בינאריים בין התקנים.

- **שכבה שנייה: שכבת הקו** - שכבת הקו בפרוטוקול Bluetooth אחראית על ניהול החיבורים ועל בקרת קישור נתונים. היא יוצרת ומחזיקה קשרים יציבים בין התקני הפרוטוקול. תפקידה הוא לבצע את כל תהליך החיבור: גילוי, אימות, הצפנה ותיקון שגיאות. השכבה יוצרת פקטות ומטפלת בבקרת זרימה ובשידור חוזר של פקטות פגומות. שכבת הקו מנהלת את צריכת החשמל, ואחראית על בחירת מצב החיבור של ההתקן.

בשכבת הקו קיים גם פרוטוקול **L2CAP**. פרוטוקול L2CAP (שעומד כקיצור ל - Logical Link Control and Adaptation Protocol) הוא פרוטוקול שתפקידו לספק ערוצים לוגיים לריבוי נתונים בין פרוטוקולים ויישומים. הפרוטוקול אחראי על פיצוח פקטות והרכבה מחדש ומשא, ומאפשר העברת נתונים בין התקני Bluetooth שונים. בכך, מאפשר גם שימוש בפרופילים ופרוטוקולים שונים בתוך Bluetooth. הפרוטוקול תומך בנוסף בריבוי פרוטוקולים על חיבור Bluetooth יחיד.

- **שכבה שלישית: השכבות העליונות** - שאר התהליך שכולל בתוכו יצירות, ניהול וסגירת שיחות בין שני התקני Bluetooth שונים, יצירת הקשר עצמו ועל שליחת המידע בתהליך, תהליך קביעת הפרופילים, הגדרת הדרישות והתנאים לגילוי ויצירת קשר כל שנותר במטרה ליצירת קשר אחר נבנה בשאר השכבות במודל שבע השכבות.



[סיכום שכבות הפרוטוקול והפרופילים מתוך המקור הנ"ל]

[2] יחידת מידה לתדירות - מיליארד פעמים בשנייה.
[3] תדרי רדיו שבהם מותר לציבור לבצע שידור וקליטה חופשיים ללא רישיון, למרחקים קצרים ובעוצמה נמוכה.
[4] שיטה לפיזור הספקטרום של אותות משודרים על ידי מיתוג מהיר של תדר המשדר כך שידלג בקצב גבוה בין מספר רב של תדרים.

תהליך הסריקה (פרוטוקול SDP)

כאשר מכשיר רוצה לגלות מכשירי Bluetooth אחרים, הוא שולח הודעה שהוא מחפש מכשירים. מכשירי Bluetooth בסביבה שרואים את ההודעה יכולים להגיב לו שהם קיימים. לאחר שאנו יודעים על המכשירי Bluetooth הקיימים בסביבתינו, נרצה לדעת מידע עליהם, כגון מהו שם המכשיר, מה המכשיר עושה, וכו'.

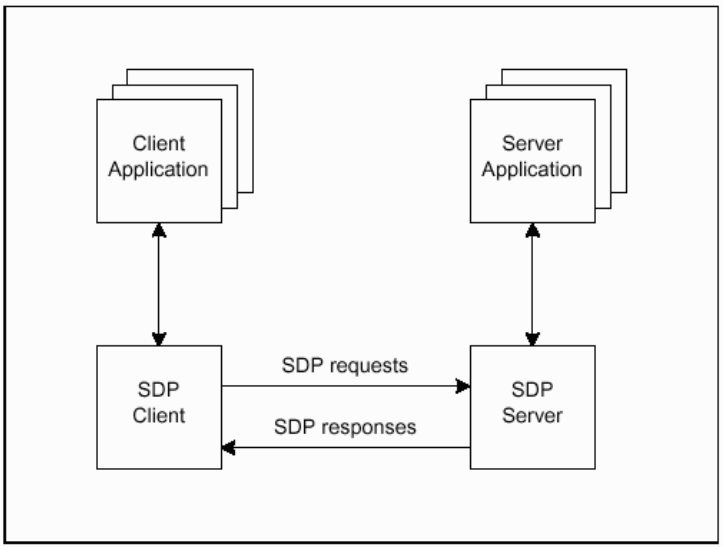
תהליך זה מתבצע ע"י פרוטוקול בשם Service Discovery Protocol, או בקצרה - SDP. פרוטוקול זה אחראי על קבלת המאפיינים של המכשירים בסביבה. מבנה הפרוטוקול בנוי בצורת Client-Server: כל מכשיר Bluetooth שרוצה שיגלו אותו מריץ שרת SDP שמכיל את ה-services שהמכשיר מציע ואת המאפיינים שלהם. כאשר אנו רוצים למצוא מידע על services שבמכשיר, נוכל לבקש את המידע בעזרת SDP client.

ישנן שתי דרכים לבקש את המידע על ה-services שבשרת:

- Browsing: דרך זו מבקשת את כל ה-services שהשרת מפרסם.
- Searching: דרך זו מבקשת את כל ה-services שהשרת מפרסם שמכילים מאפיינים מסויימים. לדוגמא, בעזרת דרך זו נוכל למצוא את כל ה-services של הדפסה שנמצאים במכשיר

כדי להדגים את מבנה ההודעה, נראה הודעה לדוגמא ב-Wireshark:

הודעה זו נשלחה ממחשב נייד כתשובה לבקשת Attribute Search, המבקשת את ה-Attributes של המכשיר. נשים לב בהודעה זו ניתן לראות את ה-features והתכונות של המכשיר, ואת כל המאפיינים שלו.



[תהליך הסריקה בשימוש בפרטוקול SDP מתוך המקור הנ"ל]

תהליך ה-Pairing

בהינתן שני התקני Bluetooth מחוברים, במידה ושני המכשירים חולקים קשר קבוע, ניתן לקשר אותם זה לזה באמצעות תהליך ה-pairing. המשמעות של התהליך הזה היא שמאותו הרגע בו המכשירים מתחברים זה לזה (באמצעות תהליך החיבור), מהפעמים הבאות תהליך החיבור הופך לאוטומטי - ללא צורך באינטראקציות של המשתמש מול הממשק. בתהליך ה-pairing, המכשירים שקושרו חולקים את הכתובות שלהם, השמות שלהם והפרופילים שלהם ובדרך כלל מאחסנים אותם בזיכרון. דוגמאות להתקנים שיכולים להתחבר בתהליך כזה הן מכשיר נייד ורכב, מכשיר נייד ואוזניות, מחשב ואוזניות וכן הלאה.

```
Bluetooth SDP Protocol
    PDU: Service Search Attribute Response (0x07)
    Transaction Id: 0x0001
    Parameter Length: 593
    Attribute List Byte Count: 590
    Data Fragment
    Continuation State: no (00)
  [Reassembled Attribute List]
    Attribute Lists [count = 16]
      Data Element: Sequence uint16 1595 bytes
          0011 0... = Data Element Type: Sequence (6)
          .... .110 = Data Element Size: uint16 (6)
          Data Element Var Size: 1595
        Data Value
          Attribute List [count =  9] (Generic Access Profile)
          Attribute List [count =  6] (Generic Attribute Profile)
          Attribute List [count =  6] (Device Information)
          Attribute List [count =  8] (A/V Remote Control)
          Attribute List [count =  8] (A/V Remote Control Target)
          Attribute List [count =  7] (Headset)
          Attribute List [count =  8] (Handsfree Audio Gateway)
          Attribute List [count =  7] (Audio Sink)
          Attribute List [count =  7] (Audio Source)
          Attribute List [count =  8] (Message Notification Server)
          Attribute List [count = 10] (Message Access Server)
          Attribute List [count =  9] (Phonebook Access Server)
          Attribute List [count =  7] (IrMC Sync)
          Attribute List [count =  7] (OBEX File Transfer)
          Attribute List [count =  8] (OBEX Object Push)
          Attribute List [count =  6] (CustomUUID: Unknown)
```

כיצד נוצרים הקשרים?

יצירת ה-pairing מחייבת ברוב ההתקנים תהליך אימות, שבו המשתמש נדרש לאמת את החיבור בין המכשירים. התהליך שונה בעיקר בין שני סוגי מכשירים שונים:

- מכשירים ללא ממשק משתמש - במכשירים מהסוג הזה, בגלל שממשק המשתמש אינו קיים לרוב אין דרישה לפעולה אקטיבית מצד המשתמש מלבד לחיצה על כפתור. לדוגמה, בחיבור אוזניות המשתמשות בפרוטוקול לטלפון, המשתמש לא נדרש לשום פעולה מלבד לחיצה על כפתור שמחבר לבד את המכשירים זה לזה. החל באותו רגע, נוצר הזוג והתהליך יתבצע באופן אוטומטי.

- מכשירים עם ממשק משתמש - במכשירים מהסוג הזה, תהליך ההתאמה בדרך כלל דורש אימות פיזי שמצריך פעולה אקטיבית מצד המשתמש ומפתח משותף לשני המכשירים, כדוגמת קוד מספרי. לדוגמה, בתהליכי התאמה עדכניים, נהוג להשתמש בקוד בן 6 ספרות ואילו בתהליכי התאמה ישנים

יותר, היה נהוג להזין קוד PIN משותף בכל מכשיר. באופן כללי, הקוד יכול לנוע באיזה אורך ומורכבות שהמכשיר ידרוש, החל בארבעה מספרים ועד מחרוזות אלפאנומריות בנות 16 תווים.

ארכיטקטורה

פרוטוקול Bluetooth מתבסס על הקמת רשת אד הוק (רשת בלתי מנוהלת שבה כל הצרכנים (מחשבים, מכשירים סלולריים התומכים בפרוטוקול וכן הלאה) מתקשרים בינם לבין עצמם ללא תשתית מאורגנת. הרשת קצרת טווח (נכון לגרסה 4.0 סביב עשרה מטרים) ומאפשרת חיבור קל וזריז בין התקנים הנמצאים זה ליד זה ללא חיבור פיזי ומבלי צורך בעזרים או התקני צד שלישי). התקן רגיל בפרוטוקול מכיל משדר או מקלט רדיו, יחידת בקרת קישוריות, יחידת תמיכה בניהול ותוכנה המאגדת בתוכה את כל הממשקים של הפרוטוקול.

פרופילים

פרופילים בפרוטוקול Bluetooth הם קבוצת כללים וחוקים המגדירים כיצד מכשירים המשתמשים בפרוטוקול יכולים לתקשר זה עם זה ולפתוח באינטראקציה חדשה. אותו סט של כללים קובע את היכולות של מכשירים ואת אפשרויות התמיכה שלהם עבור כל שימוש, ובזכות אלו ניתן ליצור שפה גלובלית לכל המכשירים התומכים בפרוטוקול למרות השוני בסוג המכשיר או היצרן.

פרופיל של מכשיר מתאר את סט הנהלים והתבניות הדרוש עבור יישומים, תוכניות או שירותים ספציפיים. כל אחד מהפרופילים מתמקד בפונקציה מסוימת וספציפית בה המכשירים יכולים להשתמש כאשר אלו מחוברים באמצעות הפרוטוקול.

בעת יצירת חיבור בין שני מכשירים שונים באמצעות הפרוטוקול, מתנהל "משא ומתן" ובו נקבע באילו פרופילים שניהם יתמכו. בכך, שני המכשירים יוכלו לדעת מה היכולות והאפשרויות של השותף לקשר ולתקשר זה עם זה באופן יעיל. ברגע בו המכשירים מחליטים על פרופיל משותף, הם יכולים לפצוח בקשר ולהחליף נתונים בצורה אפקטיבית.

סוגי הפרופילים

במהלך סעיף זה, נציג מספר פרופילים מרכזיים ומשמעותיים בפרוטוקול Bluetooth. חשוב לציין כי לא נביא את כל הפרופילים הקיימים, מספר הפרופילים בפרוטוקול הוא עצום ורובם המוחלט נמצאים בשימוש נמוך. את הרשימה המלאה ניתן למצוא כאן:

פרופיל שלט רחוק (A2DP)	**מאפשר הזרמת אודיו בין מכשירים, דוגמת אוזניות אלחוטיות, רמקולים ומערכות שמע לכלי רכב**
פרופיל הפצת אודיו (AVRCP)	מאפשר שליטה בהזרמת נתונים כגון אודיו ווידאו במכשירים רחוקים, כדוגמת אוזניות, מערכות שמע וטלוויזיות
פרופיל דיבורית (HFP)	מאפשר שיחות ללא שימוש בידיים ומאפשר למכשירים כמו אוזניות, דיבוריות רכב וטלפונים חכמים ליצור אינטראקציה לשיחות קוליות
פרופיל אוזניות (HSP)	מאפשר פונקציונליות בסיסית לתקשורת שמע וקול בין מכשירים, אשר משמשת בעיקר עבור אוזניות אלחוטיות
פרופיל התקן ממשק אנושי (HID)	מאפשר תקשורת אלחוטית עם התקני קלט כמו מקלדות, עכברים, בקרי משחק ושלטים רחוקים
פרופיל דחיפה של אובייקט (OPP)	מאפשר החלפה של אובייקטי נתונים שונים, כגון אנשי קשר, תמונות וקבצים, בין מכשירים התומכים ב-Bluetooth
פרופיל העברת קבצים (FTP)	מאפשר ומקל על העברת קבצים בין מכשירים, משמש לעתים קרובות להחלפת קבצים בין טלפונים חכמים ומחשבים
פרופיל port סדרתי (SPP)	תומך בתקשורת סדרתית בין מכשירים, המשמשת לעתים קרובות עבור יישומים כמו מדפסות אלחוטיות, סורקי ברקוד וציוד תעשייתי
פרופיל אזור רשת אישי (PAN)	מאפשר למכשירים ליצור רשתות אד-הוק לשיתוף נתונים, גישה לאינטרנט ושירותי רשת אחרים
פרופיל מכשיר בריאות (HDP)	מאפשר חילופי נתונים הקשורים לבריאות בין מכשירים. בשימוש נפוץ עבור שעוני כושר, מדי דופק ומכשירים רפואיים.
פרופיל גישה להודעות (MAP)	מאפשר חילופי הודעות, כגון SMS ודוא"ל, בין מכשירים
פרופיל גישה לספר טלפונים (PBAP)	מאפשר גישה למידע ב"ספר הטלפונים" (פרטי הקשר) במכשיר מחובר
פרופיל הפצת אודיו / וידאו כללי (GAVDP)	מספק מסגרת להזרמת אודיו ווידאו כללי בין מכשירים
פרופיל גישה כללי (GAP)	מגדיר את הדרישות הבסיסיות לגילוי מכשירי Bluetooth, יצירת חיבור ונהלי אבטחה
פרופיל תכונה גנרי (GATT)	מאפשר חילופי נתונים ופקודות בין התקני Bluetooth. בשימוש נפוץ ביישומי BLE

פרוטוקולים נוספים

מבוא לפרוטוקול Bluetooth Low Energy

בפרוטוקול Bluetooth יתרונות רבים, אך אחד החסרונות הבולטים ביותר בו הוא כמות האנרגיה שהוא צורך. כדי לטפל בבעיה הזו, ולהגיע לפתרון דומה לפרוטוקול אך עם צריכת אנרגיה נמוכה יותר (במטרה להאריך חיי סוללה ותוחלת חיים), נבנה פרוטוקול Bluetooth low energy, שקרוי גם Bluetooth 4.0 ומכונה בקצרה BLE. ההבדל העיקרי והמרכזי בין הפרוטוקולים הוא שב-BLE המצב הדיפולטי הוא לא מצב פעיל, אלא מצב שינה - כלומר, כל עוד לא נוצר חיבור המצב יהיה מצב שינה. בפועל, הפרוטוקול משמש לחיבורים קצרים יותר שלא דורשים העברת נתונים קבועה וממושכת.

Bluetooth vs BLE

מתוך ההבדל המרכזי בפעולה בין שני הפרוטוקולים, נובעים עוד הבדלים רבים שמציבים יתרונות וחסרונות לכל אחת מן השיטות:

- צריכת הסוללה: היתרון הגדול ביותר של BLE - ובכלל, כל מטרתו - הוא הצריכה הנמוכה של הנתונים ומכך גם חיסכון תמידי בסוללה. חיי הסוללה של ה-BLE ארוכים משל ה-Bluetooth ועובדה זו מהווה יתרון ל-BLE.

- קצב העברת נתונים: ל-Bluetooth קצב העברת נתונים גבוה משל BLE, כאשר הראשון עומד על כ-2.1 Mbps[5], בעוד ל-Bluetooth Low Energy יש קצב העברה של עד 1 Mbps. בשל ההבדלים בתכלית הפרוטוקולים, ל-Bluetooth קצב העברת נתונים גבוה יותר כיוון שהוא מכוון להעברת נתונים שוטפת ורציפה. לעומתו, ה-BLE נועד לצמצם כמה שיותר את צריכת האנרגיה ולתמוך בחיי הסוללה, והורדת קצב הנתונים מהווה חלק מרכזי בכך.

- Latency Rates[6] : ל-Bluetooth Low Energy יש Latency Rate נמוך יותר. Bluetooth משדר נתונים מהר יותר, אך מגיב לקלט לאט יותר; Bluetooth Low Energy משדר נתונים לאט יותר, אך מגיב לקלט מהר יותר. זה מתקיים כיוון ש-Bluetooth מיועד בעיקר למטרות שידור נתונים, ואילו BLE מתוכנן בחלקו לשימושים מעבר לשידור הפשוט של נתונים.

[5] היכולת להעביר מיליון ביטים בשנייה (מגה בייט לשנייה).
[6] מדד לכמה זמן לוקח להגיב לבקשה או לקלט.

כאן אפשר לראות השוואה ביניהם:

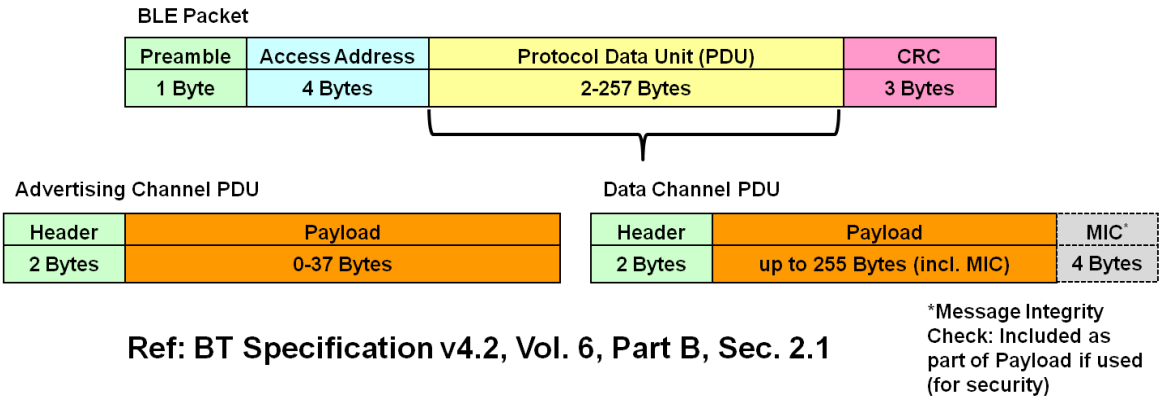
התקני Bluetooth

- אוזניות ואוזניות אלחוטיות
- רמקולים אלחוטיים
- מקלדות ומדפסות אלחוטיות
- העברת קבצים
- דיבוריות לרכב
- נקודות גישה

התקני Bluetooth low energy

- ניטור תעשייתי
- ניטור גלוקוז ולחץ דם
- ניטור בריאות הצרכן
- שעונים חכמים
- אפליקציות לתחבורה ציבורית

באיור הבא רואים את מבנה הפקטה של פרוטוקול BLE, כאשר המידע המועבר נמצא ב-Payload של השדה PDU:



[מקור: מבנה ההודעה בפרוטוקול מתוך המקור הנ"ל]

WiFi vs Bluetooth

פרוטוקול Bluetooth ו-WiFi דומים זה לזה בשל השימוש הטכנולוגי בגלי מיקרו לצורך תקשורת, ואף לעיתים חופפות. אם כך, בשביל מה בכלל צריך את פרוטוקול Bluetooth, ומה השוני ביניהן?

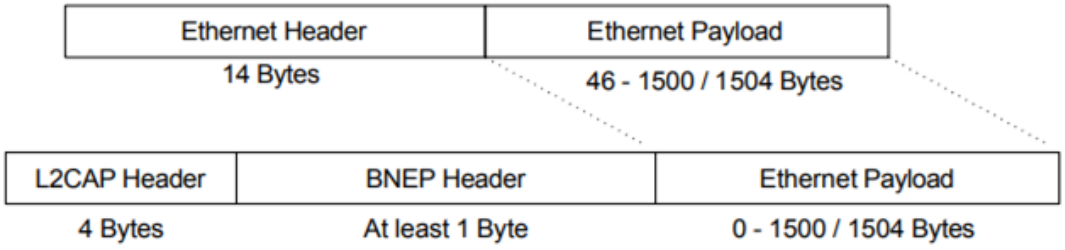
- פרוטוקול Bluetooth מתאפיין בעיקר בחיבור של מספר התקנים קרובים ברשת אישית (רשת בה אדם אחד עושה שימוש בכל רכיביה (PAN)), ומכך קצב התעבורה בה נמוך שכן ישנה העברת מידע בכמות מוגבלת. לעומתו, Wi-Fi הוא קישור ברמה מקומית (LAN) ולכן צורך הרבה יותר אנרגיה ופועל בקצב תעבורה ורוחב פס גבוהים יותר בהרבה. לכן, כל אחד מהם משמש לצורך סוג העברת נתונים שונה.

- פרוטוקול Bluetooth הומצא במטרה לספק תחליף לכבלים ישנים כפי שהסברנו במבוא, ולכן משמש לקישור במרחק מאוד קצר. לעומתו, Wi-Fi משדר בעוצמה חזקה הרבה יותר ומכך מגיע לטווחים גדולים יותר - אך גם דורש צריכת אנרגיה גבוהה יותר. בכך, Wi-Fi משמש כפרוטוקול תקשורת אלחוטית שיכול לחבר בין מכשירים מרוחקים.

אינטרנט מעל Bluetooth

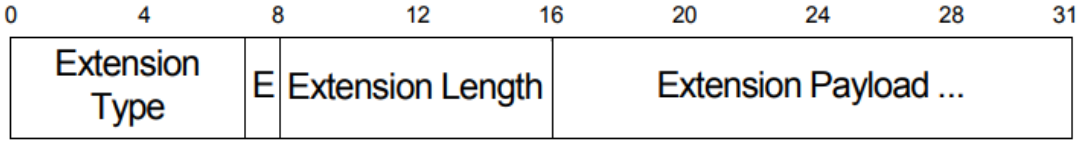
בפרוטוקול Bluetooth קיים תת-פרוטוקול שמאפשר גישה לאינטרנט מעל Bluetooth. תת-פרוטוקול זה הנקרא Bluetooth Network Encapsulation Protocol (או בקצרה - BNEP) מאפשר למכשירים לתקשר בעזרת פרוטוקולים של אינטרנט, גם אם אינם מחוברים באינטרנט. שימושים של פרוטוקול זה מכלילים הבאת גישה לאינטרנט למכשירים ללא חיבור Ethernet או Wi-Fi, ושימוש בפרוטוקולים של האינטרנט, היכולים להיות יותר קלים לשימוש ולפיתוח לחלק מהאנשים.

פרוטוקול זה עובד על ידי העברת פקטות של Ethernet מעל חיבור Bluetooth, כפי שניתן לראות באיור הבא:



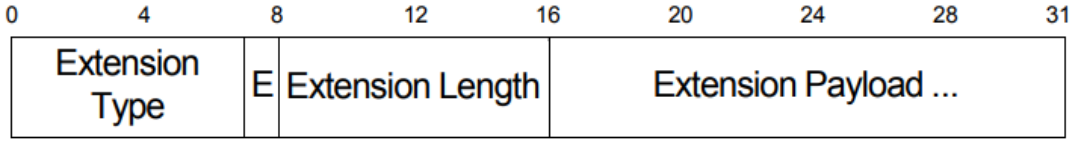
[מפרט BNEP, גרסה 1.0, עמוד 13]

בנוסף להעברת פקטות אינטרנט, BNEP מאפשר גם להעביר מידע של יצירת Personal Access Network, ושל הודעות אחרות בשם Control Messages. ה-Control Messages אחריות על שינויים וקונפיגורציה של המצב העכשווי, כגון יצירת חיבור ועוד. המבנה פקטה של ה-Control Message נראה כך:



[מפרט BNEP, גרסה 1.0, עמוד 39]

ל-Control Messages יש אפשרות להעביר עוד מידע בנוסף לשני בתים המוקצים לתוכן הפקטה, וזאת באמצעות Control Extensions. הרעיון הוא פשוט - כאשר ה-flag של ה-Extension דולק (מסומן כ-'E' באיור), תשלח עוד פקטה מסוג Extension שתכיל את המידע הנוסף. מבנה ה-Extension מוגדר כך:



[מפרט BNEP, גרסה 1.0, עמוד 39]

נשים לב שגם ל-Extension קיים flag של Extension, ויש לו את אותו התפקיד כמו בהודעת ה- Control Message - וזה מאפשר העברה בלתי מוגבלת בגודל של מידע.

למידע נוסף ומפורט יותר על הפרוטוקול ניתן לקרוא את המפרט הרשמי של הפרוטוקול.

פרוטוקול Link Manager (LMP)

פרוטוקול זה נמצא בשכבת הקו, והוא אחראי על פתיחת הקשר בין שני מכשירים, אימות, הצפנה, קינפוג הקו, ועוד.

כל הודעה בפרוטוקול נקראת Protocol Data Unit (PDU). כך נראה מבנה ה-PDU:

TID	OpCode	Parameter 1
Parameter 2		Parameter 3
...		Parameter N

ה-OpCode מכיל את סוג הבקשה, וה-TID אומר מי יצר את הבקשה. נראה מספר PDUs נפוצים:

Switch of Master-Slave Role - לעיתים, רשת מכשירי Bluetooth זקוקה להחלפה של ה-master באחד ה-slaves. תהליך החלפת התפקידים עובד בצורה הבאה: המכשיר שיזם את החלפת התפקידים מסיים את השידור של הודעת ה-L2CAP הנוכחית ולאחר מכן שולח הודעה מסוג LMP_switch_req.

אם ה-slave הוא זה שיצר את תהליך ההחלפה, תחילה תישלח הודעה מסוג LMP_slot_offset ורק לאחר מכן LMP_switch_req. במידה וההחלפה מאושרת, ה-master מסיים את השידור של הודעת ה-L2CAP הנוכחית ולאחר מכן מגיב ב-LMP_accepted. לאחר התהליך הנ"ל התפקידים מוחלפים וכעת ה-slave הוא ה-master החדש.

Pairing - כאשר שני מכשירים רוצים להתחבר, הם מתחילים את תהליך ה-pairing ע"י יצירת Initialization key שנוצר בעזרת קוד PIN ומספר רנדומלי. משם, הם מתחילים את תהליך האימות.

Authentication - תהליך האימות מכיל שני מכשירים: המאמת והמאומת. התהליך מתחיל בכך שהמאמת שולח PDU המכילה מספר רנדומלי. לאחר מכן, המאומת שולח PDU המכילה את התוצאה של פונקציה התלויה במספר הרנדומלי, כתובת המאמת, ומפתח סודי. בסוף, המאמת בודק אם התגובה נכונה לפי המפתח שהם חולקים.

Power Control - מכשיר Bluetooth יכול לבקש מהמכשיר השני לשנות את צריכת החשמל, כדי לשפר את החיבור או לחסוך חשמל. בעת קבלת הבקשה, המכשיר השני מוריד\מעלה את רמת צריכת החשמל ברמה אחת.

מבוא לאבטחה

מבוא לשינויים באבטחה

כחלק לפני גרסה 2.1, האבטחה ב-Bluetooth הייתה מינימלית: שימוש בהצפנה לא היה חובה, וגם אם משתמשים בהצפנה, אפשר להשתמש באותו מפתח רק לכ-23 שעות (זמן יותר ארוך מזה ייתן לתוקפים אפשרות להשיג את המפתח בעזרת מתקפות XOR). בנוסף, ניתן לבטל את ההצפנה בכל רגע, ומכיוון שקיימים תהליכים שדורשים ביטול הצפנה, לא ניתן לדעת האם ביטול ההצפנה נגרם מהתקפת אבטחה או מתהליכים רגילים.

גרסה 2.1 פותרת את הבעיות האלו, בכך שהיא מכריחה את השימוש בהצפנה מתי שאפשר, ופעולות המחייבות ביטול הצפנה מודיעות על כך לפני בעזרת פיצ'ר חדש בשם Encryption Pause and Resume. בנוסף, מפתח ההצפנה חייב להתעדכן לפני שלא יהיה אפשר להשתמש בו יותר. את טבלת הגרסאות והשינויים המלאים ניתן לראות בנספח בסוף המאמר.

מבנה אבטחת המידע

מבנה האבטחה של פרוטוקול Bluetooth מתפתח ומשתנה בין גרסה לגרסה, בהתאם לשינויים ולפרצות אבטחה שמתרחשות בפרוטוקול. באופן כללי, אבטחת הפרוטוקול מושתתת על חמישה עקרונות עיקריים: אימות, סודיות, pairing, הרשאות ווידוא לגבי שלמות מסרים.

על כל אלו הסברנו כבר בחלקו השני של המאמר ועל הצורה בה הם מתממשים. פרוטוקול Bluetooth מסתמך בנוסף על ארבעה אופני אבטחה שקובעים את דרגת האבטחה:

דרגה	תקן
1	ללא אבטחה - ללא הצפנה, אימות וכו'. לא נמצא בשימוש כלל.
2	האבטחה הנתונה עד גרסה 2.0 - השירות הנתון לפי ההסבר הקודם. נמצא בשימוש רק במכשירים ישנים ולא מעודכנים.
3	קישור פיזי בין מכשירים - אבטחה פיזית, עם כל התכונות הדרושות - אך עם פתח לתקיפות שמנצלות את תהליך האימות הפיזי כדי לגשת דרכו לצמתים אחרים ברשת הפרוטוקול.
4	האבטחה הנתונה החל בגרסה 2.1 - השירות הנתון לפי ההסבר הקודם.

מיון וסיווג פרצות אבטחה

לפני שנתחיל להתמקד בפרצות אבטחה, חשוב להבין שישנו סיווג מאוד ברור שקובע מה סוג כל מתקפת אבטחה ומאפשר אבחון קל יותר ומיקוד דרכי הטיפול. אלו הם סוגי המתקפות:

- **מתקפת Bluejacking** - מתקפת אבטחה מהסוג הזה בדרך כלל כוללת השתלטות לא רצויה על תקשורת ה-Bluetooth מצד מכשיר אחד, ודרכה העברת הודעות לא רצויות (כדוגמת הפצת תמונות לא רצויות, פרטים אישיים, פרסומות וכו').

- **מתקפת Bluesmacking** - מתקפת אבטחה מהסוג הזה משמשת לביצוע התקפות מניעת שירות (DoS[7]). מתקפות מהסוג הזה פועלות בצורה בה נשלחת כמות מידע גדולה מדי למכשיר היעד באמצעות (L2CAP) במטרה להציף את הבקרה של הקשר. המכשיר קורס ונאלץ לכבות. ניתן לשחזר את הקשר על ידי אתחול מחדש, אך התקפות מהסוג הזה משמשות כשער להתקפות חמורות הרבה יותר שיכולות להתרחש כאשר המכשיר אינו פעיל.

- **מתקפת Bluesnarfing** - מתקפת אבטחה מהסוג הזה דומה במהותה למתקפות Bluejacking, אך קיים ביניהם הבדל משמעותי שהופך את Bluesnarfing לחמורה הרבה יותר - במתקפה מהסוג הזה לא רק שניתן להשתלט על הקשר ועל שליחת הנתונים, אלא ניתנת גישה למידע האישי של מכשיר הפרוטוקול (כדוגמת תמונות אישיות, טקסטים, מיילים, סיסמות וכו'). בנוסף, הפורצים יכולים להשתמש במידע הנ"ל כדי לנתב מחדש כתובות, פרטי לוח שנה, פרטי בנק או שיחות טלפון למכשירים התוקפים ובכך להשתלט על המידע. הפירצה מנצלת נקודות תורפה בפרוטוקול Object Exchange[8].

- **מתקפת Bluebugging** - מתקפת אבטחה מסוג זה היא כאשר הפורץ משתמש בחיבור בפרוטוקול בכדי להשתלט על מכשיר היעד של הפריצה ולהתקין תכונות זדוניות. התוכנות הללו מעניקות לפורץ אפשרויות רבות (כדוגמת יזימת שיחות טלפון, גישה לעריכת פרטי קשר, קריאת ושליחת הודעות וכו').

- **מתקפת Location Tracking** - מתקפת האבטחה הזו מתייחסת בעיקר ל-BLE ומצביעה על אפשרות למעקב אחרי מיקום בשימוש בפרוטוקול הנ"ל במכשירים שמשתמשים במיקום. הפורצים יכולים לנצל את אותות הפרוטוקול במטרה לעקוב אחרי המיקום של הנתקף בזמן אמת (כדוגמת שעונים חכמים וכו').

---

[7] מתקפות שנועדו להשבית מערכת מחשב על ידי עומס חריג על משאביה.
[8] פרוטוקול תקשורת המאפשר החלפת אובייקטים בינאריים בין מכשירים.

מבוא לפרצת BlueBorne

אחת מפרצות האבטחה המוכרות מהשנים האחרונות היא BlueBorne. Blueborne הינה אוסף של 8 פרצות אבטחה שממוקדות בפלטפורמות שונות (Android, IOS, Linux ו-Windows) שהתגלו ע"י חברת Armis ודווחו ב-12 בספטמבר, 2017. לפי ההערכות 5.3 מיליארד מכשירים היו בסכנה מהפרצות האלו ב-2017, וגם ב-2018 - לאחר שנה, העריכו שעדיין היו יותר מ-2 מיליארד מכשירים בסכנה מהפרצות האלו.

בחלק זה נעבור על 4 מהפרצות האלו.

פרצת BlueBorne: פרוטוקול SDP

במערכות ההפעלה Android ו-Linux, היו פרצות אבטחה בהשמה של פרוטוקול הסריקה SDP. תחילה, נפרט על החלק בו היו הפריצות בפרוטוקול:

כאשר לקוח שולח לשרת ה-SDP בקשה, השרת SDP עונה לו. אם התשובה של השרת SDP עולה על הגודל המקסימלי המוגדר לפקטה, השרת שולח רק חלק מהבקשה שמתאים בתוך הגודל המקסימל ומוסיף לפקטה הזו מידע בשם Continuation State. כאשר הלקוח רוצה לראות את המשך הפקטה, הוא שולח Continuation Request לשרת, שמכילה את ה-Continuation State שקיבל מהשרת. כשהשרת מקבל את הבקשה, הוא שולח לו את המשך הפקטה. אם המשך הפקטה עולה על הגודל המקסימלי, השרת שוב שולח רק חלק מהפקטה עם Continuation State כפי שהוסבר קודם לכן. ה-Continuation State משתנה בין שרת לשרת, ואין לו מפרט קבוע.

כעת, נסתכל על הפירצה ב-Linux:

ה-Continuation State ב-Linux מכילה את הגודל של התשובה של השרת:

```
typedef struct {
        uint32_t timestamp;
        union {
                uint16_t maxBytesSent;
                uint16_t lastIndexSent;
        } cStateValue;
} sdp_cont_state_t;
```

לכן, משתמש זדוני יכול לשלוח Continuation Request עם Continuation State שמכיל גודל תשובה גדול יותר ממה שהשרת נתן, והשרת ידליף מידע שנמצא ב-heap. מידע זה יכול להכיל מפתחות הצפנה שנמצאים בשימוש ב-Bluetooth, וכך יכול לאפשר מתקפת המשך.

הפירצה ב-Android דומה:

ה-Continuation State ב-Android מכילה את ה-offset שממנו צריך להמשיך לשלוח את התשובה. לקוראים שיודעים שפת C ומעוניינים, מצורפת ההגדרה של ה-Continuation State:

```
typedef struct {
    uint16_t cont_offset;
} sdp_cont_state_t;
```

אמנם ב-Android מתבצעת בדיקה על ה-offset הזה, עדיין היה ניתן לבצע פריצה:

משתמש זדוני יוכל לשלוח שתי בקשות לשרת: בקשה אחת שמחזירה הרבה מידע, ובקשה אחרת שמחזירה קצת (אך עדיין יותר מהגודל המקסימלי לפקטה). כעת, המשתמש ישלח Continuation Request עבור הבקשה השנייה (הקטנה יותר) אך עם ה-Continuation State של הבקשה הראשונה. דבר זה יוביל לבלבול בשרת של ה-Android ויגרום לו לשלוח מידע שמתחיל ב-offset לא נכון. ה-Continuation State יעבור את הבדיקות מכיוון שהוא תקין ונלקח מבקשה, ודבר זה יביא ל-underflow ולכן השרת ידליף מידע שנמצא ב-heap. כמו הפירצה ב-Linux, מידע זה יכול להכיל מפתחות הצפנה שנמצאים בשימוש ב-Bluetooth, וכך יכול לאפשר מתקפת המשך.

פרצת BlueBorne: פרוטוקול BNEP

במערכת ההפעלה Android נמצאו 2 פרצות אבטחה במימוש הפרוטוקול BNEP. הפירצה הראשונה נמצאת במקרה הקצה הבא: מכיוון שניתן לשלוח מספר Control Messages באותה פקטה, משתמש יכול לשלוח פקטה שמכילה מספר BNEP Control Messages. כאשר הוא שולח הודעה אחת של פתיחת חיבור והשאר אחרות, מכיוון שפתיחת החיבור לוקחת זמן, Android שומר את שאר ההודעות בזיכרון כדי שיטפל בהם לאחר שיסיים את פתיחת החיבור. אך כאשר Android שומר את ההודעות האחרות, כתוצאה מבאג בתוכנה הוא דורס 8 בתים בזיכרון וכותב בהם מידע מהפקטות שהוא שומר. דבר זה גורם לפירצה שנותנת למשתמש זדוני אפשרות לכתוב לזיכרון. כביכול היו אמורים לשים לב לבעיה זו קודם מכיוון שבכל פעם שהקוד הזה רץ הוא דורס חלק בזכרון, אך המקרה שהקוד מטפל בו אינו נפוץ, ולכן כנראה שהוא אף פעם לא רץ עד שמצאו את הפירצה הזו.

לקוראים שיודעים שפת C ומעוניינים, מצורף הקוד שגורם לדריסת ה-8 בתים:

```
UINT8 *p = (UINT8 *)(p_buf + 1) + p_buf->offset;
...
type = *p++;
extension_present = type >> 7;
type &= 0x7f;
...
switch (type)
{
...
case BNEP_FRAME_CONTROL:
    ctrl_type = *p;
    p = bnep_process_control_packet (p_bcb, p, &rem_len, FALSE);
    if (ctrl_type == BNEP_SETUP_CONNECTION_REQUEST_MSG &&
        p_bcb->con_state != BNEP_STATE_CONNECTED &&
        extension_present && p && rem_len)
    {
        p_bcb->p_pending_data = (BT_HDR *)osi_malloc(rem_len);
        memcpy((UINT8 *)(p_bcb->p_pending_data + 1), p, rem_len);
        ...
    }
...
```

הפירצה השנייה קשורה לטיפול של BNEP Extension Messages עבור הודעה לא ידועה. לפי הגדרת הפרוטוקול, כאשר מקבלים הודעת Extension Message לא מוכרת, צריך להתעלם ממנה.

במקרה של Android, כאשר שולחים Extension Message עבור Control Message לא מוכרת, Android מתעלם מההודעה הזו ומוריד מכמות הבתים שנשארו בפקטה את אורך המידע ב-Extension Message (נקרא במבנה הפקטה Extension Length).

אך קיימת כאן בעיה - אין כאן בדיקה שה-Extension Length הוא אכן נכון ושזה הגודל של שארית ההודעה. לכן, משתמש זדוני יכול לשלוח הודעת Extension לא מוכרת עם אורך לא נכון, ולגרום ל-underflow במשתנה המחזיק את כמות הבתים שנותרו לפקטה. בעזרת ה-underflow, משתמש זדוני יכול להמשיך את המתקפה ולכתוב מידע לזיכרון.

לקוראים שיודעים שפת C ומעוניינים, מצורף הקוד שמכיל את הפירצה:

```
...
if (is_ext)
{
    ext_len = *p++;
    *rem_len = *rem_len - 1;
}
...
control_type = *p++;
*rem_len = *rem_len - 1;
...
switch (control_type)
{
...
default :
    ...
    if (is_ext)
    {
        p += (ext_len - 1);
        *rem_len -= (ext_len - 1);
    }
    break;
}
...
```

ניתן לראות הדגמה של ניצול הפרצות האלו ב-Android כאן. אם ברצונכם ללמוד עוד על BlueBorne, אנו ממליצים לקרוא את ה-Paper White על הנושא. כל התמונות בחלק זה נלקחו מה-*White Paper* המקושר למעלה.

BrakTooth

פרצת אבטחה נוספת שגילו באוגוסט 2021 היא BrakTooth. כמו BlueBorne, הפירצה BrakTooth היא בעצם אוסף של פרצות במגוון מכשירים שונים הממוקדות במימושים של Bluetooth שנמצאים ב-firmware של המכשיר, ומשפיעות על מעל 1400 מכשירים מיצרנים שונים:

Product Vendor	Product Type	Product Model
Microsoft	Laptop	Surface Laptop 3 Surface Go 2 Surface Pro 7 Surface Book 3
Dell	Desktop PC / Laptop	Optiplex 5070 Alienware M17 R3 **(Many more)**
Sony	Smartphone	Xperia XZ2
Oppo	Smartphone	Reno 5G CH1921
Ericsson	Home Entertainment Hub	KDE20102
Volvo Technology	Automotive Infotainment	Volvo FH **(Many More)**
Hella GmbH	Electronic Control Unit	PMP3
Walmart Stores	Audio	Disco Lamp Speaker **(Many More)**
Walmart Stores	Audio	Small rugged speaker **(Many More)**
Panasonic	Audio	Sound Bar SC-HTB100
Becker Avionics	Aircraft Entertainment System	AMU6500 **(More)**
u-box	Industrial IoT Module	NINA-W106
Koyo Electronics	PLC (Industrial Automation)	C2-02CPU

[רשימה של חלק מהמכשירים המושפעים מ-BrakTooth, מתוך אתר group research ASSET]

מכיוון ש-firmware של מכשירים זה דבר שבדרך כלל לא מפרסמים את קוד המקור שלו, לא נוכל לעבור על הפרצות האלה באופן מלא כמו שעשינו ב-BlueBorne.

בכל זאת, נעבור על שתי פרצות מעניינות: הפירצה הקריטית ביותר נמצאת ב-firmware של chip בשם ESP32, שמשתמשים בו בעיקר בתחום ה-IoT. בעקבות חוסר בדיקה של out-of-bounds בספריית ה-Bluetooth של המכשיר, ניתן לרשום שמונה בתים של מידע ובכך לדרוס קריאה לפונקציה בקריאה לפונקציה אחרת. כתוצאה מכך, תוקף זדוני היודע את ה-firmware של המכשיר יכול לקרוא לפונקציות לא רצויות ובכך להשיג שליטה רבה על המכשיר. ניתן לראות הדגמה של פירצה זו בקישור זה.

פירצה נוספת שקיימת משפיעה על מכשירים המשתמשים ב-Intel AX200 או ב-Qualcomm WCN3990, ביניהם מכשירי לפטופ וסמארטפונים. מתקפה זו יכולה לגרום ל-Denial of Service, ולגרום להפסקת החיבור בין מכשירי Bluetooth או להפרעות בתקשורת ביניהם.

ניתן לראות הדגמה של פירצה זו בקישור זה. אם ברצונכם ללמוד עוד על BrakTooth, אנו ממליצים להסתכל באתר של Group Asset או לקרוא את ה-Paper White על הנושא.

סיכום

במאמר זה סקרנו את ההיסטוריה ותהליך הפעולה של חלק מפרוטוקולי Bluetooth על קצה המזלג, תיארנו את מבנה האבטחה של הפרוטוקולים ובחנו פרצות ידועות. בחלקו הראשון של המאמר, התחלנו בלהבין את הצורך בפרוטוקול ואת ההיסטוריה שלו, ולאחר מכן את הרקע התיאורטי שלו, תוך רכישת מושגים בסיסיים בפרוטוקול כמו מצבי החיבור ופרופילים ותהליכים כמו תהליך החיבור וה-Pairing.

סקרנו וריאנטים שונים של הפרוטוקול כמו BLE ואינטרנט מעל Bluetooth והשוונו אותם לפתרונות אחרים כמו WiFi.

בחלקו השני של המאמר סקרנו את מבנה האבטחה של פרוטוקול Bluetooth ופרצות אבטחה בו. התחלנו מלעבור על מבנה האבטחה ועל היסטורית השינויים בה, ולאחר מכן סיווגנו פרצות אבטחה לסוגי מתקפות שונות. לבסוף, בעזרת הכלים שרכשנו חקרנו את פרצות BlueBorne ו-BrakTooth לעומק ובחנו את השפעותן.

אודות

איתמר שבתאי עמיעזר, בן 16 מפתח תקווה ו**דניאל שוסטק**, בן 17 מתל אביב. תלמידי תוכנית אודיסאה במסלול הסייבר שנה ג׳ במרכז מדעני העתיד, במסגרת אוניברסיטת תל אביב לנוער.

המאמר נכתב במסגרת פרויקט כתיבת מאמרים בתוכנית, וברצוננו להודות לד"ר **שלומי בוטנרו** על ההנחיה בפרויקט ובכתיבת המאמר.

ביבליוגרפיה

- https://he.m.wikipedia.org/wiki/בלוטות%27
- Bluetooth Basics - SparkFun Learn
- Bluetooth for Programmers
- Bluetooth - GeeksforGeeks
- The Bluetooth Protocol Stack
- The Bluetooth Protocol Architecture
- Bluetooth Protocol: Overview and Bluetooth Module - Latest Open Tech From Seeed
- Bluetooth Protocol (Part 1): Basics and Working
- Bluetooth Protocol – Type, Data Exchange and Security
- What Is BLE (Bluetooth Low Energy) and How Does It Work?
- Bluetooth Low Energy (BLE): A Complete Guide
- https://www.Bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=556599
- The Bluetooth Standard - A simple guide to the protocol for beginners
- Bluetooth Basics
- BlueTooth Protocol - PiEmbSysTech
- Bluetooth Pairing: Network Connection » Electronics Notes
- https://www.amd.e-technik.uni-rostock.de/ma/gol/lectures/wirlec/Bluetooth_info/sdp.html
- Bluetooth Tutorial
- BlueBorne
- BrakTooth