

פרויקט באבטחת מידע 236349

חולשות אבטחה בתקן (WPS) Wi-Fi Protected Setup

מגישות: אנה נובין, מיטל ברכה סבו

מנחה: עמיחי שולמן

תוכן עניינים

3	הקדמה
4	רקע
4	מטרת הפרויקט
5	כלים וסביבות פיתוח
5	Wireshark
5	כרטיס רשת
5	Scapy
6	VMware Workstation
6	Access Point
6	מכשירי קצה
7	סקר ספרות ותוצאות קודמות
7	KRACK Attack
7	KARMA and Evil Twin Attacks
8	SSID striping
8	Wi-Fi Protected Setup (WPS) Vulnerable to Brute-Force Attack
10	מהלך הפרויקט
10	מחקר מבנה ההודעות בתקן WPS
12	מחקר קוד פתוח של תקן WPS
33	Fuzzer
41	לוח זמנים
41	לוח זמנים הנקבע בדו"ח הפתיחה
41	לוח הזמנים בפועל
42	סיכום והצעות לשיפור

הקדמה

כיום, במקומות רבים, וביניהם במקומות עבודה, במוסדות להשכלה גבוהה ובבתים פרטיים מותקנת רשת תקשורת אלחוטיות מקומית (Wi-Fi) המבוססת על תקן 802.11. היתרונות המרכזיים של רשת מסוג זה הינם פשטות החיבור של נקודות קצה לרשת ומחירה הזול.

אחת הדרכים בהן באה פשטות זו לידי ביטוי הינו תקן ה-WPS (Wi-Fi Protected Setup). תקן זה מאפשר להוסיף נקודות קצה חדשות לרשת בלחיצה על כפתור ה-WPS בנקודת גישה, ללא הזנת סיסמא ע"י המשתמש של נקודת הקצה. תקן זה אמנם מקל על המשתמשים, אך להערכתנו גם הופך אותם ואת הרשת עצמה לפגיעים להתקפות שונות.

בפריקט זה נתמקד בחקירת תקן זה, ובבדיקת היתכנות של חולשות אבטחה שונות הנוצרות כתוצאה משימוש בו. נבדוק חולשות הן בנקודות קצה והן במכשירי המשתמשים.

רקע

תקן ה-WPS הינו תקן אבטחת תקשורת המשמש להקמת רשת תקשורת אלחוטית מקומית (Wi-Fi). תקן זה מובנה בדרך כלל בנקודות גישה כדי להקל על התחברות מכשירי קצה ועל הוספת מכשירי קצה חדשים לרשת ה-Wi-Fi באמצעות נקודות גישה אלה.

ההתחברות בתקן זה מתבצעת בלחיצת כפתור הנמצא על נקודת הגישה או בהזנת PIN, המאפשרים התחברות ללא סיסמא.

התקן מורכב מהחלפת מידע בין מכשיר הקצה המעוניין להתחבר לרשת לנקודת הגישה, המועבר באמצעות information element שמתווסף להודעות ה-beacon, response, probe request ו-authentication request\ response, ולאחר מכן באמצעות סדרה של הודעות EAP.

מטרת הפרויקט

מטרת הפרויקט הינה מחקר וחשיפה של פרצות אבטחה שונות בתקן (WPS) Wi-Fi Protected Setup. מטרת התקן WPS הינה להקל על הקמת רשת אלחוטית ביתית המוגנת בפרוטוקול אבטחה WPA או WPA2, וכן להקל על הוספת תחנות קצה חדשות לרשת.

במסגרת הפרויקט נתבונן לעומק בהודעות הראשונות העוברות בפרוטוקול, באמצעות מעבר על קוד פתוח המממש את הפרוטוקול במערכת ה-Linux. נתמקד בעיקר בהודעות מסוג WSC IE (מסוג TLV), ובבחן נקודות תורפה בבדיקות ובפעולות המתבצעות בהן. זאת במטרה לנצל נקודות תורפה אלה ולגרום למכשיר הקצה או לנקודות גישה להתנהגות לא רצויה, כגון קריסה, הזלגת מידע, הרצת קוד ועוד.

לשם כך, נבנה fuzzer אשר יבחן באופן שיטתי את חוזקת הפרוטוקול, בהתאם לנקודות התורפה שמצאנו בחקר הקוד הפתוח, ואת הדרכים בהן ניתן לנצל אותן כדי לתקוף את מכשירי הקצה וכן את נקודות הגישה.

כלים וסביבות פיתוח

במהלך הפרויקט נעזרנו בכלים הבאים:

Wireshark



תוכנה מבוססת קוד פתוח המאפשרת להאזין ולהקליט בזמן אמת תקשורת מחשבים העוברת בנקודה כלשהי ברשת אליה נגיש המחשב עליו מותקנת התוכנה. ההאזנה מתבצעת בעזרת כרטיס רשת הנמצא ב-monitor mode המחובר למחשב עליו מותקנת התוכנה.

מטרתה העיקרית לשמה עשינו שימוש ב-Wireshark הינה ניתוח עמוק של תעבורת רשת באמצעות צפייה במבנה והתוכן של ההודעות שהוקלטו, הן כדי ללמוד על תקן WPS והן כדי לבדוק את הצלחת ה-fuzzer.

כרטיס רשת

חומרת מחשב המאפשרת למחשב להתחבר לרשת מחשבים. כתובת ה-MAC של כרטיס הרשת משמשת את המחשב בכל התקשורת שעוברת דרך אותו כרטיס רשת.

במהלך הפרויקט, הפעלנו את כרטיס הרשת ב-monitor mode, המאפשר למחשב להאזין לכל התקשורת אליה הוא נגיש. הקלטנו תקשורת זו באמצעות התוכנה Wireshark, וכן שלחנו באמצעות כרטיס הרשת הודעות שבנינו במסגרת ה-fuzzer.

Scapy



תוכנה מבוססת פייתון המאפשרת לבצע מניפולציות על פקטות, ובמסגרת כך לשלוח, לתפוס, לפרק ולבנות פקטות במגוון פרוטוקולי תקשורת. יכולות אלה, מאפשרות למשתמש לבנות כלים שמטרתם הן לבחון רשתות תקשורת למציאת פריצות אבטחה והן לתקוף רשתות אלה באמצעות פריצות האבטחה שנמצאו.

במהלך הפרויקט, השתמשנו בתוכנה זו בשביל לבנות fuzzer הבוחן נקודות תורפה אפשריות בהודעות הנשלחות במסגרת תקן WPS.

VMware Workstation



תוכנה המאפשרת הרצה של מערכת הפעלה במסגרת מכונה וירטואלית.

במהלך הפרויקט, השתמשנו בתוכנה זו בשביל לעבוד במערכת הפעלה Linux המאפשרת עבודה נוחה ויעילה יותר עם הכלים בהם נעזרנו.

Access Point

במהלך הפרויקט השתמשנו ב-Wi-Fi Router מסוג Edimax N300 לשם ניתוח ההודעות העוברות בעת התחברות מכשיר קצה לנקודת הגישה בתקן WPS. בנוסף, השתמשנו במכשיר זה בשביל לבחון את השפעת הבדיקות שביצענו באמצעות ה-fuzzer על נקודת הגישה.

מכשירי קצה

במהלך הפרויקט השתמשנו במכשירי קצה העומדים לרשותנו כגון טלפונים סלולריים, עם מערכת הפעלה IOS ו-Android ומחשבים עם מערכת הפעלה Windows.

סקר ספרות ותוצאות קודמות

לאורך השנים נבחנו התקפות שונות על פרוטוקול ה-Wi-Fi. סקירת הספרות שלנו התמקדה במציאת מתקפות שונות ובחיפוש אחר כיוונים שטרם נסו. נפרט מעט על חלק מהמתקפות שקראנו עליהן:

KRACK Attack

המתקפה הזו מתרכזת ביכולת לקרוא את כל התקשורת בין מכשיר הקצה לנקודת גישה. המתקפה לא מקנה יכולת לגלות סיסמאות של המשתמש, אך היא כן מאפשרת לגלות את מפתח ההצפנה של session תקשורת ספציפי על ידי האזנה והתערבות ב-four way handshake.

התוקף במקרה הזה מנצל יכולות MitM, הוא מאזין לתקשורת של לחיצת הידיים, אבל חוסם את ההודעה הרביעית (זו שסוגרת את הפרוטוקול). כעת נקודת הגישה תאלץ לשלוח שוב את הודעה 3 מכיוון שמעולם לא התקבל עליה ACK. כאשר המכשיר המותקף מקבל שוב את הודעה 3, הוא מתקין מחדש את כל מפתחות ההצפנה, ומאתחל את המשתנים הקשורים כגון ה-nonce.

התגלית המעניינת במתקפה הזו, היא שמפתח ההצפנה אינו משתנה בשליחה החוזרת של הודעה 3. מצידו של המכשיר המותקף, לאחר שליחת הודעה 4, הוא מתחיל לשלוח הודעות data. התוקף מאזין ושומר הודעות אלו. בכל פעם שמפתח ההצפנה נשלח מחדש, ה-nonce מאותחל גם הוא, ונשלחות חבילות data עם אותו nonce ואותו מפתח הצפנה. מכאן, המסלול עבור התוקף למצוא את מפתח ההצפנה כבר יותר פשוטה, בעיקר כיוון שיש ביכולתו להשיג כמה הודעות כאלה שירצה.

מכיוון שמתקפה זו תוקפת את פרוטוקול ה-Wi-Fi עצמו, ולא מימוש ספציפי, כותבי המאמר מצאו שהיא יעילה כנגד כל מכשיר שבו מומש הפרוטוקול באופן תקין. המתקפה הזו יעילה כנגד כל מימוש של WPA2 וכן WPA ו-AES שקדמו לו.

קישור: <https://www.krackattacks.com>

KARMA and Evil Twin Attacks

שתי המתקפות הללו מנצלות את רשימת ה-PNL (preferred network list) שמכשירים תומכי Wi-Fi מחזיקים אצלם. זוהי רשימה של רשתות שהמכשיר התחבר אליהן בעבר ויהיו מוכן להתחבר אליהן שוב ללא התערבות המשתמש. זיהוי הרשתות ברשימה זו נעשה על פי ה-SSID של הרשת, כך שרשת זדונית, המחזיקה באותו SSID כמו אחת הרשתות ברשימה הנ"ל, תוכל לגרום למכשיר המותקף להתחבר אליה באופן אוטומטי ללא צורך בהתערבות המשתמש.

ההבדל בין שתי המתקפות הוא באופן בחירת ה-SSID, מתקפת Evil Twin מגרילה SSID אקראי, בעוד מתקפת KARMA מרחיכת אחד הודעות broadcast ששולח המכשיר המותקף, הודעות אלו מכילות את רשימת ה- PNL באופן לא מוצפן, כך שהתוקף יכול פשוט לבחור שם מתוך הרשימה ולבצע את המתקפה.

קישור: https://en.wikipedia.org/wiki/KARMA_attack

SSID stripping

מתקפה נוספת שמנצלת את ה-SSID. במתקפה הזו, שהתגלתה בשיתוף פעולה עם סטודנטים בטכניון ניתן להפעיל מניפולציות על הודעת ה-broadcast, כך שהשם האמיתי של הרשת וזה שמוצג למשתמש שונים זה מזה.

במכשירים רבים קיימת שכבת הגנה שמונעת התחברות אוטומטית לרשתות עם שמות לא מזהים, אך כותבי המאמר מצאו שכאשר הם כוללים ברצף הביטים של ה-SSID ביטים שאים מזהים עם תו printable (כגון NULL) הם מצליחים לגרום לשלושה סוגי שגיאות- השמטה של חלק משם הרשת האמיתי, שינוי סדר הופעת הרשתות הזמינות במכשיר ודחיפת רשתות אל מחוץ לרשימה המוצגת (overflow).

חולשות אלו עזרו לכותבי המאמר לגרום למשתמשי הקצה לראות שם שבטחו בו ולנסות להתחבר לרשת הזדונית.

קישור: <https://aireye.tech/2021/09/13/the-ssid-stripping-vulnerability-when-you-dont-see-what-you-get/>

Wi-Fi Protected Setup (WPS) Vulnerable to Brute-Force Attack

התראה שפורסמה ע"י National Cyber Awareness System בשנת 2012 מתארת חולשה בתקן ה-WPS. החולשה נובעת מתכנון לקוי של התקן, ומקטינה באופן משמעותי את מספר הסיסמאות האפשריות בהן משתמש WPS לשם ביצוע authentication בין מכשיר הקצה לנקודת הגישה. התוקף יכול לנצל חולשה זו בהתקפה הבאה: תוקף הנמצא בטווח רדיו יכול לגלות את WPS PIN של נקודת גישה באמצעות brute-force. באופן זה, התוקף יכול לקבל לידייו את סיסמאות ה-WEP או ה-WPA ולקבל בסבירות גבוהה גישה לרשת ה-Wi-Fi. התקפת ה-brute-force על WPS PIN מאפשר לגלות את הסיסמא תוך 4-10 שעות.

הפתרון לחולשה זו הינו השבתת האפשרות לחיבור ב-WPS או לחלופין עדכן ה-firmware במידה ויצרן נקודת הגישה פרסם עדכון המטפל בחולשה זו.

קישור: [https://www.cisa.gov/uscert/ncas/alerts/TA12-](https://www.cisa.gov/uscert/ncas/alerts/TA12-006A#:~:text=An%20attacker%20within%20radio%20orange,traffic%20and%20mount%20fur)

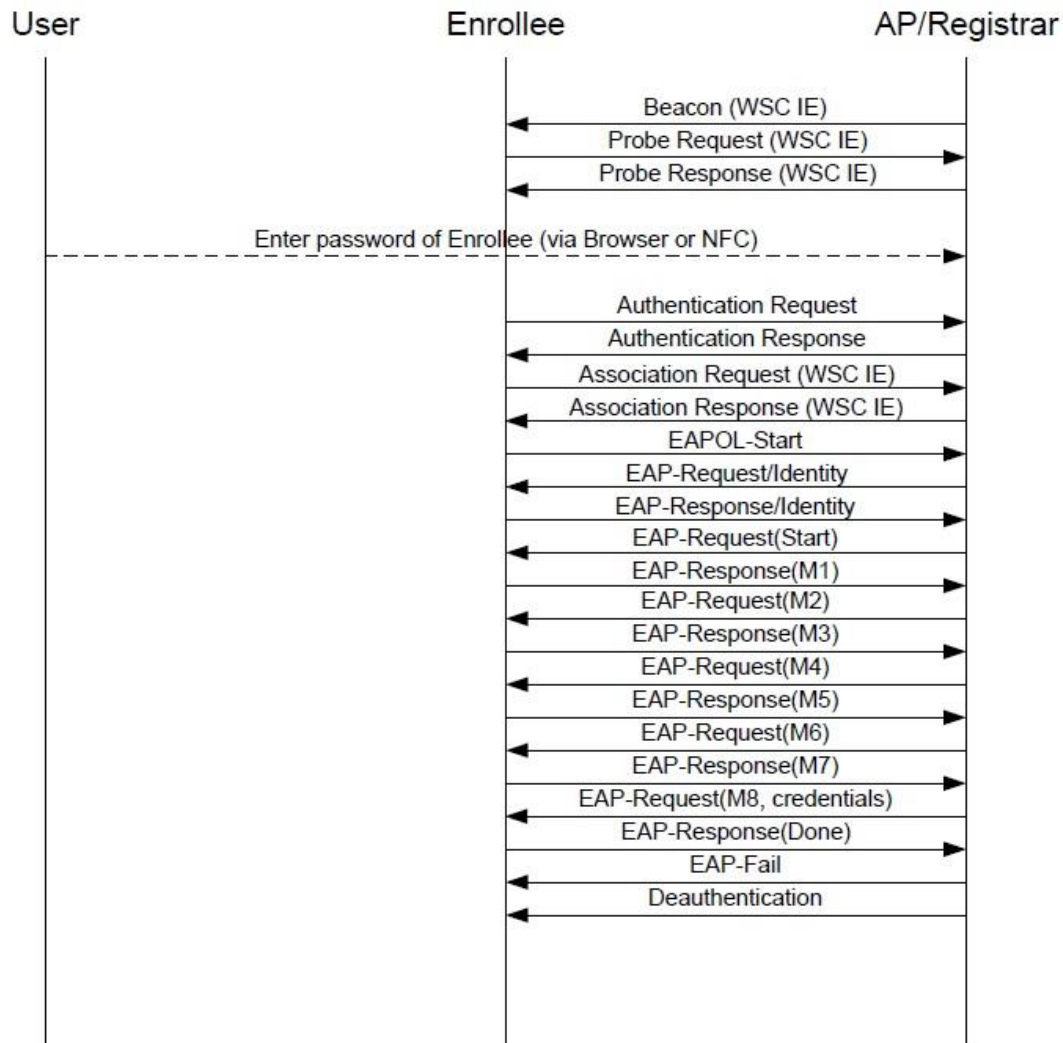
006A#:~:text=An%20attacker%20within%20radio%20orange,traffic%20and%20mount%20fur
.ther%20attacks

מהלך הפרויקט

מחקר מבנה ההודעות בתקן WPS

בשלב הראשון, התעמקנו בהודעות העוברות בעת התחברות מכשיר קצה לנקודת גישה בתקן ה-WPS. בשלב זה, הבנו את סדר ההודעות ואת המבנה שלהן. ביצענו זאת, במספר דרכים.

ראשית, מצאנו את התרשים הבא, המתאר את סדר ההודעות:



כפי שניתן לראות, שלושת ההודעות הראשונות נשלחות עוד לפני שנעשתה התערבות כלשהי בתקן ה-WPS. חלק זה של התקן הוא החלק שראינו בו התעסקות רבה בסקר הספרות שעשינו על מתקפות קיימות.

התבוננו על חלק מהודעות הפרוטוקול יותר מקרוב. בכל הודעה שבחנו, השדות שעיינינו אותנו היו שדות מסוג TLV (Type Length Value).

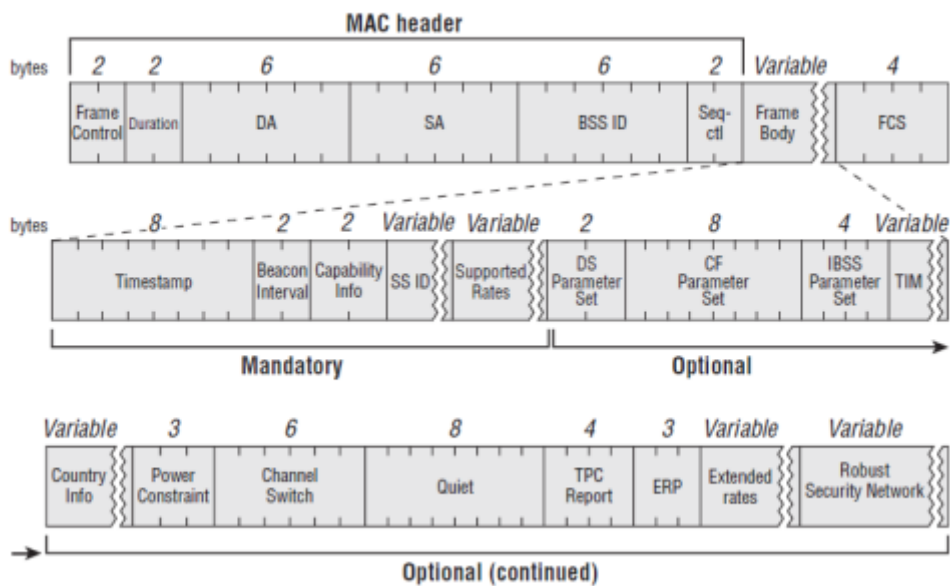
שדות אלה מורכבים משלושה חלקים (כנרמז בשמם):



Figure 4-31. Generic management frame information element

לדוגמא בהודעת ה- Beacon, הבנויה באופן הבא:

FIGURE 4.5 Beacon frame structure



חייבים להופיע בין היתר השדות הבאים:

1. SSID - שדה זה מופיע בכל beacon, response\ probe request ו- association request response. ה- type של שדה זה הינו 0, ה- subtype הינו 8 ואורכו מוגבל ל- 32 תווים.
2. Supported Rates - שדה זה מופיע בכל הודעה שבה מופיע ה- SSID ובנוסף גם ב- association request\ response. שדה זה מורכב מאלמנטים שאורך כל אחד מהם הינו 8 ביטים, וכל אלמנט כזה מייצג תדר נתמך אחד. הביט ה- 7 בכל אלמנט מייצג האם התדר הזה הוא מסוג "בסיסי או חובה".

מופיעים בנוסף השדות: Country, FH parameter set, Extended Supported Rates, RSN – Robust Security Network, BSS Load, Secure Network, Vendor Specific Fields ועוד.

מחקר קוד פתוח של תקן WPS

בשלב השני, מצאנו קוד מקור פתוח המממש נקודת גישה התומכת ב-WPS וחקרנו קוד זה כ- white box.

המימוש אותו חקרנו נמצא בקישור: web.mit.edu/freesbd/head/contrib/wpa/src/wps/.

מטרת מחקר הקוד הייתה לחפש נקודות תורפה פוטנציאליות באלמנטים המועברים בשיטת ה-TLV בתוך ה-IE. בשלב הראשון, מצאנו את חלק הקוד שמקבל את ההודעות הראשונות של הפרוטוקול, אלו שאחראיות להעברת הפרטים ואימות הזהויות. קטע קוד זה נמצא בקובץ

:src/wps/wps_enrollee.c

```
static enum wps_process_res wps_process_wsc_msg(struct wps_data *wps,
                                               const struct wpabuf *msg)
{
    struct wps_parse_attr attr;
    enum wps_process_res ret = WPS_CONTINUE;

    wpa_printf(MSG_DEBUG, "WPS: Received WSC_MSG");

    if (wps_parse_msg(msg, &attr) < 0)
        return WPS_FAILURE;

    if (attr.enrollee_nonce == NULL ||
        os_memcmp(wps->nonce_e, attr.enrollee_nonce, WPS_NONCE_LEN) != 0) {
        wpa_printf(MSG_DEBUG, "WPS: Mismatch in enrollee nonce");
        return WPS_FAILURE;
    }

    if (attr.msg_type == NULL) {
        wpa_printf(MSG_DEBUG, "WPS: No Message Type attribute");
        wps->state = SEND_WSC_NACK;
        return WPS_CONTINUE;
    }

    switch (*attr.msg_type) {
    case WPS_M2:
        if (wps_validate_m2(msg) < 0)
            return WPS_FAILURE;
        ret = wps_process_m2(wps, msg, &attr);
        break;
    case WPS_M2D:
        if (wps_validate_m2d(msg) < 0)
            return WPS_FAILURE;
        ret = wps_process_m2d(wps, &attr);
        break;
    case WPS_M4:
        if (wps_validate_m4(msg) < 0)
            return WPS_FAILURE;
        ret = wps_process_m4(wps, msg, &attr);
        if (ret == WPS_FAILURE || wps->state == SEND_WSC_NACK)
            wps_fail_event(wps->wps, WPS_M4, wps->config_error,
                          wps->error_indication,
                          wps->peer_dev.mac_addr);
        break;
    }
```

```

case WPS_M6:
    if (wps_validate_m6(msg) < 0)
        return WPS_FAILURE;
    ret = wps_process_m6(wps, msg, &attr);
    if (ret == WPS_FAILURE || wps->state == SEND_WSC_NACK)
        wps_fail_event(wps->wps, WPS_M6, wps->config_error,
                        wps->error_indication,
                        wps->peer_dev.mac_addr);

    break;
case WPS_M8:
    if (wps_validate_m8(msg) < 0)
        return WPS_FAILURE;
    ret = wps_process_m8(wps, msg, &attr);
    if (ret == WPS_FAILURE || wps->state == SEND_WSC_NACK)
        wps_fail_event(wps->wps, WPS_M8, wps->config_error,
                        wps->error_indication,
                        wps->peer_dev.mac_addr);

    break;

default:
    wpa_printf(MSG_DEBUG, "WPS: Unsupported Message Type %d",
               *attr.msg_type);
    return WPS_FAILURE;
}

/*
 * Save a copy of the last message for Authenticator derivation if we
 * are continuing. However, skip M2D since it is not authenticated and
 * neither is the ACK/NACK response frame. This allows the possibly
 * following M2 to be processed correctly by using the previously sent
 * M1 in Authenticator derivation.
 */
if (ret == WPS_CONTINUE && *attr.msg_type != WPS_M2D) {
    /* Save a copy of the last message for Authenticator derivation
     */
    wpabuf_free(wps->last_msg);
    wps->last_msg = wpabuf_dup(msg);
}

return ret;
}

```

בפונקציה לעיל מצאנו switch קלאסי המבדיל בין ההודעות השונות ושולח כל אחת מהן לפונקציה מפרסרת ייעודית. אבל לפני המיון לפי סוג ההודעה, הבחנו בשורה הבאה:

```

static enum wps_process_res wps_process_wsc_msg(struct wps_data *wps,
                                               const struct wpabuf *msg)
{
    struct wps_parse_attr attr;
    enum wps_process_res ret = WPS_CONTINUE;

    wpa_printf(MSG_DEBUG, "WPS: Received WSC_MSG");
    → if (wps_parse_msg(msg, &attr) < 0)
        return WPS_FAILURE;
}

```

פונקציה הזו מקבלת את תוכן ההודעה שנשלחה לפני שעברה כל וידוא או בדיקה. כלומר, כל הבדיקות הנדרשות מתבצעות על ידי הפונקציה עצמה. הפונקציה נמצאת בקובץ `src/wps/wps_attr_parse.c` בשורה 596.

בחנו את המסלול שעוברת ההודעה בפונקציה, תוך מתן דגש על מציאת בדיקות שעובר הבאפר.

בשלב הראשון מתבצע אתחול של מספר משתנים:

```
int wps_parse_msg(const struct wpabuf *msg, struct wps_parse_attr *attr)
{
    const u8 *pos, *end;
    u16 type, len;
    #ifdef WPS_WORKAROUNDS
    u16 prev_type = 0;
    #endif /* WPS_WORKAROUNDS */

    os_memset(attr, 0, sizeof(*attr));
    → pos = wpabuf_head(msg);
    end = pos + wpabuf_len(msg);
```

בקטע הקוד לעיל לא קיימת בדיקה של התוכן, אך שמנו לב שבשורה המסומנת אורך ההודעה נבדק על ידי הקוד המקבל ולא על פי מה שדווח על ידי השולח. מכאן שמתקפה המנסה לדווח על אורך הודעה שונה מזה שנשלח במציאות, כגון מתקפה שמדווחת על הודעה ארוכה יותר על מנת לגרום לתוכנה לקרוא ולהדליף זיכרון שלא אמורה להיות לנו גישה אליו תיפול בשלב הזה.

שאר ההודעה מורכב מלולאת while גדולה שרצה כל עוד pos נמצא בתוך הבאפר.

```
while (pos < end)
```

קידום הלולאה נעשה בסופה והוא נראה כך:

```
#ifdef WPS_WORKAROUNDS
    prev_type = type;
#endif /* WPS_WORKAROUNDS */
    pos += len;
}
```

המשתנה len מקבל את ערכו בתוך הלולאה מתוך שדה ה-len שמגיע כחלק ממבנה ה-TLV.

כעת ניגשנו לנתח את גוף הלולאה עצמו. הקוד מורכב מארבעה תנאים וקטע קוד קצר שמתבצע בכל מקרה. בתנאי הראשון נעשית בדיקה בסיסית של אורך הבאפר, אם אורכו הכולל הוא בביתים שלמים

```
if (end - pos < 4) {
    wpa_printf(MSG_DEBUG, "WPS: Invalid message - "
               "%lu bytes remaining",
               (unsigned long) (end - pos));
    return -1;
}
```

בדיקה זו לכאורה חוסמת, בשילוב עם תנאי העצירה של הלולאה, את האפשרות לפגיעה בעזרת אורך הודעה הכולל. אבל כפי שראינו קודם, pos הוא משתנה שמעודכן על פי len המדווח על ידי התוקף. לכן רשמנו נקודה זו כנקודת תורפה פוטנציאלית.

לאחר התנאי הראשוני, הגיע קטע הקוד שאיננו מותנה, זהו למעשה קטע המרכזי של הפונקציה:

```

type = WPA_GET_BE16(pos);
pos += 2;
len = WPA_GET_BE16(pos);
pos += 2;
wpa_printf(MSG_EXCESSIVE, "WPS: attr type=0x%x len=%u",
           type, len);

```

מכיוון ש-`type` ו-`len` באלמנטים של TLV מוגדרים להיות באורך של שני בתים, הקוד כאן פשוט לוקח את שני הערכים האלה ישירות מהבאפר. כאן נמצאת נקודת התורפה של `len`, מה שידווח על ידי התוקף ילקח כפי שהוא. המשמעות היא שאם נכניס כאן ערכים אסטרטגיים נוכל להשפיע על סיום הלולאה. לכאורה, אם נדווח כאן על `len` שלילי נוכל לגרום ללולאה לרוץ לנצח. אבל כאשר חזרנו להסתכל בתחילת הפונקציה, מצאנו ש `len` מוגדר כ-`unsigned` כך שפרצה זו חסומה.

המשכנו לקטע המותנה הבא:

```

if (len > end - pos) {
    wpa_printf(MSG_DEBUG, "WPS: Attribute overflow");
    wpa_hexdump_buf(MSG_MSGDUMP, "WPS: Message data", msg);
#ifdef WPS_WORKAROUNDS
    /*
     * Some deployed APs seem to have a bug in encoding of
     * Network Key attribute in the Credential attribute
     * where they add an extra octet after the Network Key
     * attribute at least when open network is being
     * provisioned.
     */
    if ((type & 0xff00) != 0x1000 &&
        prev_type == ATTR_NETWORK_KEY) {
        wpa_printf(MSG_DEBUG, "WPS: Workaround - try "
                  "to skip unexpected octet after "
                  "Network Key");
        pos -= 3;
        continue;
    }
#endif /* WPS_WORKAROUNDS */
    return -1;
}

```

הבדיקה כאן חוסמת את אחת מנקודות התורפה של אלמנטים מסוג TLV, התוכנה בודקת כאן אפשרות שבה `len` שדווח ארוך מזה שנשלח. מתקפה נפוצה מאוד היא לדווח על אורך ארוך מזה שנשלח ובכך לגרום לתוכנה להדליף מידע שלא אמור להקרא. הבדיקה שמבוצעת כאן חוסמת אפשרות זו. הקטע הבא בקוד אינו מתבצע עבור כל קימפול של הקוד. קטע זה נועד להתמודד עם ריפוד של אפסים. הקוד מוצא את הבית הראשון שאינו 0 וממשיך משם.

```

#ifdef WPS_WORKAROUNDS
    if (type == 0 && len == 0) {
        /*
         * Mac OS X 10.6 seems to be adding 0x00 padding to the
         * end of M1. Skip those to avoid interop issues.
         */
        int i;
        for (i = 0; i < end - pos; i++) {
            if (pos[i])
                break;
        }
        if (i == end - pos) {
            wpa_printf(MSG_DEBUG, "WPS: Workaround - skip "
                "unexpected message padding");
            break;
        }
    }
#endif /* WPS_WORKAROUNDS */

```

גם כאן זיהינו חולשה אפשרית, ריפוד של אפסים יכול להזיז את pos למיקום שאינו בכפולה של זוגות בתים. נניח שהערך הבא שמופיע הוא 0x1230 ה- Workaround שמופיע כאן יגרום לכך שה- type הבא שיקרא יהיה 0x123_ כאשר המספר האחרון יהיה למעשה התו הראשון של שדה ה- len, וה- len יכיל תווים מתוך ה- value.

השורה האחרונה שנותרה לנו לחקור היא הקריאה לפונקציה הבאה בשרשרת הקריאות

```

if (wps_set_attr(attr, type, pos, len) < 0)
    return -1;

```

מצאנו את הפונקציה הזו באותו הקובץ, בשורה 152. הפונקציה מכילה switch-case גדול על כל התכונות האפשריות שיכולות להופיע בשדה ה- type. בסוף הפונקציה מופיע ה- default הבא:

```

default:
    wpa_printf(MSG_DEBUG, "WPS: Unsupported attribute type 0x%x "
        "len=%u", type, len);
    break;
}

```

Case זה חוסם את נקודת התורפה של הזרקת type לא מוגדר- התכנה פשוט תתעלם ממנו. החולשה היחידה באופן טיפול המוצג היא הדפסה של ערכים מהתוקף אל תוך ה- error log. המסקנה מכאן, היא שכל מתקפה חייבת לעשות שימוש ב- ATTR המוכרים על ידי התכנה.

רשימת ה- enum המוגדרים נמצאת בקישור:

https://w1.fi/cgiit/hostap/tree/src/wps/wps_defs.h#n55 ומכילה את כל הערכים ההקסדצימליים

בטווח 0x1001-0x1064 ועוד שני ערכים מחוץ לטווח 0x106a, 0x10fa.

המבנה של כל case דומה, הוא מתחיל בבדיקות על ה- len, אם יש, ולבסוף מעתיק מצביע אל הביטים בבאפר שדווחו ב- len אל מבנה הנתונים ATTR ללא בדיקה נוספת על התוכן.

כעת ניגשנו לחלק המרכזי של המחקר בו בחנו לעומק מקרים בהם הבדיקות על המשתנה len בסיסיות מדי או כלל לא קיימות, על מנת לזהות עוד נקודות תורפה.

המקרים שבחנו הם:

- Line 284:

```
case ATTR_OOB_DEVICE_PASSWORD:
    if (len < WPS_OOB_PUBKEY_HASH_LEN + 2 ||
        len > WPS_OOB_PUBKEY_HASH_LEN + 2 +
            WPS_OOB_DEVICE_PASSWORD_LEN ||
        (len < WPS_OOB_PUBKEY_HASH_LEN + 2 +
            WPS_OOB_DEVICE_PASSWORD_MIN_LEN &&
            WPA_GET_BE16(pos + WPS_OOB_PUBKEY_HASH_LEN) !=
            DEV_PW_NFC_CONNECTION_HANDOVER)) {
        wpa_printf(MSG_DEBUG, "WPS: Invalid OOB Device "
            "Password length %u", len);
        return -1;
    }
    attr->oob_dev_password = pos;
    attr->oob_dev_password_len = len;
    break;
```

- Line 459:

```
case ATTR_MANUFACTURER:
    attr->manufacturer = pos;
    if (len > WPS_MANUFACTURER_MAX_LEN)
        attr->manufacturer_len = WPS_MANUFACTURER_MAX_LEN;
    else
        attr->manufacturer_len = len;
    break;
```

- Line 466:

```
case ATTR_MODEL_NAME:
    attr->model_name = pos;
    if (len > WPS_MODEL_NAME_MAX_LEN)
        attr->model_name_len = WPS_MODEL_NAME_MAX_LEN;
    else
        attr->model_name_len = len;
    break;
```

- Line 473:

```
case ATTR_MODEL_NUMBER:
    attr->model_number = pos;
    if (len > WPS_MODEL_NUMBER_MAX_LEN)
        attr->model_number_len = WPS_MODEL_NUMBER_MAX_LEN;
    else
        attr->model_number_len = len;
    break;
```

- Line 480:

```
case ATTR_SERIAL_NUMBER:
    attr->serial_number = pos;
    if (len > WPS_SERIAL_NUMBER_MAX_LEN)
        attr->serial_number_len = WPS_SERIAL_NUMBER_MAX_LEN;
    else
        attr->serial_number_len = len;
    break;
```

- Line 487:

```
case ATTR_DEV_NAME:
    if (len > WPS_DEV_NAME_MAX_LEN) {
        wpa_printf(MSG_DEBUG,
            "WPS: Ignore too long Device Name (len=%u)",
            len);
        break;
    }
    attr->dev_name = pos;
    attr->dev_name_len = len;
    break;
```

- Line 513:

```
case ATTR_ENCR_SETTINGS:
    attr->encr_settings = pos;
    attr->encr_settings_len = len;
    break;
```

- Line 517:

```
case ATTR_CRED:
    if (attr->num_cred >= MAX_CRED_COUNT) {
        wpa_printf(MSG_DEBUG, "WPS: Skipped Credential "
            "attribute (max %d credentials)",
            MAX_CRED_COUNT);
        break;
    }
    attr->cred[attr->num_cred] = pos;
    attr->cred_len[attr->num_cred] = len;
    attr->num_cred++;
    break;
```

- Line 537:

```
case ATTR_NETWORK_KEY:
    attr->network_key = pos;
    attr->network_key_len = len;
    break;
```

לאחר שמצאנו את המקרים החשודים בנקודות התורפה, חקרנו אותם אחד אחד. כל חקירה התחילה מחיפוש על פני כל קבצי המערכת של המקומות בהם נעשה שימוש במשתנה שלתוכו נשמר הערך או המצביע. לאחר מכן, התחכינו אחר שרשרת הקריאות שעושה שימוש במשתנה על מנת לזהות משתנים שעוברים את הפעפוע הגדול ביותר לתוך הקוד. מתוך מחשבה, שככל שיש יותר שימושים (ללא בדיקות נוספות) יש יותר נקודות תורפה פוטנציאליות עם האלמנט.

הבדיקה הראשונה שביצענו עבור כל אחד מהמקרים ב-switch היא לחפש על פני כל הקבצים את המקומות בקוד שבהן נעשית השמה או העתקה של התוכן שהתקבל במקרים השונים. כעת נפרט את המקרים השונים שבדקנו:

ATTR_OOB_DEVICE_PASSWORD:

תוצאות החיפוש הראשוני:

```
wpa_supplicant/wps_supplicant.c
2479:     wpa_s->wps->registrar, attr->oob_dev_password,
2480:     attr->oob_dev_password_len);

src/ap/wps_hostapd.c
2006:     data.oob_dev_pw = attr->oob_dev_password;
2007:     data.oob_dev_pw_len = attr->oob_dev_password_len;

src/p2p/p2p_parse.c
505:     msg->oob_dev_password = attr.oob_dev_password;
506:     msg->oob_dev_password_len = attr.oob_dev_password_len;

src/wps/wps_attr_parse.c
296:     attr->oob_dev_password = pos;
297:     attr->oob_dev_password_len = len;
```

התוצאה הראשונה הובילה לשרשרת קריאות שבסופה מחושב `id = WPA_GET_BE16(pos);` ומושם לתוך:

```
token->peer_pk_hash_known = pubkey_hash != NULL;
if (pubkey_hash)
    os_memcpy(token->pubkey_hash, pubkey_hash,
              WPS_OOB_PUBKEY_HASH_LEN);
token->pw_id = pw_id;
token->pk_hash_provided_oob = pk_hash_provided_oob;
if (dev_pw) {
    wpa_snprintf_hex_uppercase((char *) token->dev_pw,
                               sizeof(token->dev_pw),
                               dev_pw, dev_pw_len);
    token->dev_pw_len = dev_pw_len * 2;
}
```

שרשרת הקריאות המלאה היא:

wpa_supplicant/wps_supplicant.c/[wpas_wps_add_nfc_password_token](#)

src/wps/wps_registrar.c/[wps_registrar_add_nfc_password_token](#)

src/wps/wps_registrar.c/[wps_registrar_add_nfc_pw_token](#)

בצילום המסך הבא ניתן לראות את המקומות בהן המשתנה שמצאנו נקרא מתוך הקוד:

```
wpa_supplicant/mesh_rsn.c
340:   if (sta->sae->tmp && !sta->sae->tmp->pw_id && ssid->sae_password_id) {
341:       sta->sae->tmp->pw_id = os_strdup(ssid->sae_password_id);
342:       if (!sta->sae->tmp->pw_id)

src/ap/ieee802_11.c
546:       rx_id = sta->sae->tmp->pw_id;
638:   if (!data && sta->sae->tmp && sta->sae->tmp->pw_id)

src/common/sae.c
114:   os_free(tmp->pw_id);
2011:   if (sae->tmp->pw_id) {
2014:       sae->tmp->pw_id);
2017:       os_free(sae->tmp->pw_id);
2018:       sae->tmp->pw_id = NULL;
2030:   if (sae->tmp->pw_id &&
2031:       (len != os_strlen(sae->tmp->pw_id) ||
2032:        os_memcmp(sae->tmp->pw_id, epos, len) != 0)) {
2035:       sae->tmp->pw_id);
2039:   os_free(sae->tmp->pw_id);
2040:   sae->tmp->pw_id = os_malloc(len + 1);
2041:   if (!sae->tmp->pw_id)
2043:   os_memcpy(sae->tmp->pw_id, epos, len);
2044:   sae->tmp->pw_id[len] = '\0';
2046:       sae->tmp->pw_id, len);

src/wps/wps_registrar.c
55:       if (pw_id == 0 || pw_id == token->pw_id)
66:       if (pw_id == token->pw_id)
1397:       if (wps->nfc_pw_token->pw_id == DEV_PW_NFC_CONNECTION_HANDOVER)
1919:       wps->nfc_pw_token->pw_id == DEV_PW_NFC_CONNECTION_HANDOVER) {
3720:   token->pw_id = pw_id;
```

מצאנו משתנה עם פוטנציאל, אבל לא גדול מאוד כיוון שהשימוש בו די מועט. לכן המשכנו לחפש.

עברנו לתוצאה השנייה בחיפוש המקורי, בפונקציה הראשונה מתבצעת העתקה של הביטים אל תוך משתנה של המערכת ומבאן המשתנה מפעפע דרך שרשרת הקריאות:

```
wpa_supplicant/wps_supplicant.c/wpas_wps_nfc_rx_handover_sel
wpa_supplicant/wps_supplicant.c/wpas_wps_start_nfc
wpa_supplicant/wps_supplicant.c/wpas_wps_start_dev_pw
src/utils/common.c/wpa_snprintf_hex_uppercase
```

בשורה השלישית בשרשרת הקריאות מתבצעת בדיקה שהתוכן שונה מ-0, ובקריאה האחרונה הערך מודפס ללא שימוש נוסף, לכן הגענו למסקנה שלמשתנה זה אין ערך למחקר שלנו.

כאשר התבוננו בתוצאה השלישית בחיפוש המקורי ראינו שהתוצאה נמצאת בקבצים שאינם נוגעים ל-AP ולכן אינה רלוונטית עבורנו.

ATTR_MANUFACTURER:

תוצאות החיפוש הראשוני:

```
wpa_supplicant/wps_supplicant.c
1583: wps->dev.manufacturer = wpa_s->conf->manufacturer;
2270: wps->dev.manufacturer = wpa_s->conf->manufacturer;

wpa_supplicant/ap.c
742: bss->manufacturer = os_strdup(wpa_s->conf->manufacturer);

wpa_supplicant/p2p_supplicant.c
4836: p2p.manufacturer = wpa_s->conf->manufacturer;

wpa_supplicant/config_file.c
1122: fprintf(f, "manufacturer=%s\n", config->manufacturer);

wpa_supplicant/dbus/dbus_new_handlers_p2p.c
1464: tmp = os_strdup(info->manufacturer);

src/ap/wps_hostapd.c
1127: wps->dev.manufacturer = hapd->conf->manufacturer ?

src/p2p/p2p.c
2956: p2p->cfg->manufacturer = os_strdup(cfg->manufacturer);

src/wps/wps_enrollee.c
1072: m2d->manufacturer = attr->manufacturer;

src/wps/wps_er.c
51: ev->manufacturer = sta->manufacturer;
139: evap->manufacturer = ap->manufacturer;
558: wpa_printf(MSG_DEBUG, "WPS ER: manufacturer='%s'", ap->manufacturer);
799: sta->manufacturer = dup_binstr(attr->manufacturer,

src/wps/wps_dev_attr.c
21: len = dev->manufacturer ? os_strlen(dev->manufacturer) : 0;
```

השורה הראשונה מובילה לפונקציה `wpa_supplicant/wps_supplicant.c/wpas_wps_init` שנקראת פעם אחת בקוד מתוך השרשרת הבאה:

```
wpa_supplicant/wpa_supplicant.c/wpa_supplicant_add_iface
wpa_supplicant/wpa_supplicant.c/wpa_supplicant_init_iface
wpa_supplicant/wps_supplicant.c/wpas_wps_init
```

מסלול הקריאה מוביל חזרה לקבצים שקשורים ל-WPA ולא ל-WPS שהוא נושא המחקר שלנו. אומנם נעשה שימוש בקבצים אלה עבור WPS אך החולשות שם משותפות גם ל-WPA שנחקר רבות, אז עברנו לתוצאה הבאה. למעשה, דילגנו קדימה לתוצאה שנמצאת בקובץ השייך ל-WPS בפונקציה:

```
src/ap/wps_hostapd.c/hostapd_init_wps
```

מקריאה מדוקדקת של שורות הקוד הקודמות להשמה המופיעה בצילום המסך, מצאנו כי גם כאן אין בדיקה נוספת של התוכן מעבר לבדיקת האורך הבסיסית שכבר ראינו ב-switch-case. חיפשנו את השימושים במבנה החדש שנוצר ומצאנו את שרשרת הקריאות הבאה:

```
src/wps/wps_upnp_web.c
199:     s = iface->wps->dev.manufacturer;
```

```
src/wps/wps_upnp_web.c/format_wps_device_xml
src/wps/wps_upnp_web.c/web_connection_parse_get
src/wps/wps_upnp_web.c/web_connection_check_data
```

בתיעוד הפונקציה האחרונה בשרשרת מצאנו נקודת תורפה מבטיחה:

```
/* Called when we have gotten an apparently valid http request.
 */
static void web_connection_check_data(void *ctx, struct http_request *req)
{
```

מצאנו מבנה נתונים הנקרא בכל שמתקבלת בקשת http ועלול להכיל תוכן שנשלח ע"י התוקף ולא עבר בדיקות מספקות.

ATTR_MODEL_NAME:

תוצאות החיפוש הראשוני:

```
wpa_supplicant/wps_supplicant.c
1584: wps->dev.model_name = wpa_s->conf->model_name;
2271: wps->dev.model_name = wpa_s->conf->model_name;

wpa_supplicant/ap.c
744: bss->model_name = os_strdup(wpa_s->conf->model_name);

wpa_supplicant/p2p_supplicant.c
4837: p2p.model_name = wpa_s->conf->model_name;

wpa_supplicant/config_file.c
1124: fprintf(f, "model_name=%s\n", config->model_name);

wpa_supplicant/dbus/dbus_new_handlers_p2p.c
1501: tmp = os_strdup(info->model_name);

src/ap/wps_hostapd.c
1129: wps->dev.model_name = hapd->conf->model_name ?

src/p2p/p2p.c
2958: p2p->cfg->model_name = os_strdup(cfg->model_name);

src/wps/wps_enrollee.c
1074: m2d->model_name = attr->model_name;

src/wps/wps_er.c
52: ev->model_name = sta->model_name;
142: evap->model_name = ap->model_name;
569: wpa_printf(MSG_DEBUG, "WPS ER: modelName='%s'", ap->model_name);
805: sta->model_name = dup_binstr(attr->model_name,

src/wps/wps_dev_attr.c
45: len = dev->model_name ? os_strlen(dev->model_name) : 0;
```

החיפוש הזה כמעט זהה לזה של ה-ATTR הקודם, כל תוצאות החיפוש נמצאות שורה אחת אחרי השורות מהחיפוש הקודם ועוברות את אותו הטיפול בדיוק.

ATTR_MODEL_NUMBER:

תוצאות החיפוש הראשוני:

```
wpa_supplicant/wps_supplicant.c
1585: wps->dev.model_number = wpa_s->conf->model_number;
2272: wps->dev.model_number = wpa_s->conf->model_number;

wpa_supplicant/ap.c
746: bss->model_number = os_strdup(wpa_s->conf->model_number);

wpa_supplicant/p2p_supplicant.c
4838: p2p.model_number = wpa_s->conf->model_number;

wpa_supplicant/config_file.c
1126: fprintf(f, "model_number=%s\n", config->model_number);

wpa_supplicant/dbus/dbus_new_handlers_p2p.c
1538: tmp = os_strdup(info->model_number);

src/ap/wps_hostapd.c
1131: wps->dev.model_number = hapd->conf->model_number ?

src/p2p/p2p.c
2960: p2p->cfg->model_number = os_strdup(cfg->model_number);

src/wps/wps_enrollee.c
1076: m2d->model_number = attr->model_number;

src/wps/wps_er.c
53: ev->model_number = sta->model_number;
143: evap->model_number = ap->model_number;
572: wpa_printf(MSG_DEBUG, "WPS ER: modelNumber='%s'", ap->model_number);
811: sta->model_number = dup_binstr(attr->model_number,

src/wps/wps_dev_attr.c
69: len = dev->model_number ? os_strlen(dev->model_number) : 0;
```

ניתן לראות ש-ATTR זה מופיע בדיוק באותם מקומות כמו ה-ATTR הקודם.

ATTR_SERIAL_NUMBER:

תוצאות החיפוש הראשוני:

```
wpa_supplicant/wps_supplicant.c
1586: wps->dev.serial_number = wpa_s->conf->serial_number;
2273: wps->dev.serial_number = wpa_s->conf->serial_number;

wpa_supplicant/ap.c
748: bss->serial_number = os_strdup(wpa_s->conf->serial_number);

wpa_supplicant/p2p_supplicant.c
4839: p2p.serial_number = wpa_s->conf->serial_number;

wpa_supplicant/config_file.c
1128: fprintf(f, "serial_number=%s\n", config->serial_number);

wpa_supplicant/dbus/dbus_new_handlers_p2p.c
1575: tmp = os_strdup(info->serial_number);

src/ap/wps_hostapd.c
1133: wps->dev.serial_number = hapd->conf->serial_number ?

src/p2p/p2p.c
2962: p2p->cfg->serial_number = os_strdup(cfg->serial_number);

src/wps/wps_enrollee.c
1078: m2d->serial_number = attr->serial_number;

src/wps/wps_er.c
54: ev->serial_number = sta->serial_number;
145: evap->serial_number = ap->serial_number;
578: wpa_printf(MSG_DEBUG, "WPS ER: serialNumber='%s'", ap->serial_number);
817: sta->serial_number = dup_binstr(attr->serial_number,

src/wps/wps_dev_attr.c
93: len = dev->serial_number ? os_strlen(dev->serial_number) : 0;
```

ניתן לראות שה- ATTR הנ"ל זהה לשלושת ה- ATTR-ים הקודמים. מצב שבו ארבעה ATTR רצופים עוברים טיפול זהה הינו נקודת תורפה פוטנציאלית בפני עצמה. ארבעת ה-ATTR שבדקנו עוברים טיפול זהה, על אף שהם אינם אותו אובייקט וכל אחד אמור לקיים דרישות מעט שונות. המשתנים נשמרים במבנה נתונים אחד בזיכרון ויתכן שאף בצורה סריאלית בזיכרון הפיזי. כלומר, מצאנו פוטנציאל להכניס קטע קוד ארוך יותר מדרישות האורך שנבדקות בפונקציית ה- parse. וכפי שראינו בחקר ה-ATTR הראשון מבנה הנתונים שאליו הם נשמרים נקרא פעמים רבות.

`ATTR_DEV_NAME:`

תוצאות החיפוש הראשוני:

```
doc/dbus.doxygen
1373:     <tr><td>dev_name</td><td>ay</td>

wpa_supplicant/wps_supplicant.c
794:     enrollee->dev_name ? enrollee->dev_name : "",

wpa_supplicant/dbus/dbus_new.c
699:         (const char *) m2d->dev_name,

src/p2p/p2p_parse.c
692:     cli->dev_name = (const char *) g;
754:     os_memcpy(name, cli->dev_name, cli->dev_name_len);

src/p2p/p2p.c
453:         cli->dev_name, cli->dev_name_len);
2953: if (cfg->dev_name)
2954:     p2p->cfg->dev_name = os_strdup(cfg->dev_name);
3026: os_free(p2p->cfg->dev_name);
3089: os_free(p2p->cfg->dev_name);
3091:     p2p->cfg->dev_name = os_strdup(dev_name);
3092:     if (p2p->cfg->dev_name == NULL)
3095:         p2p->cfg->dev_name = NULL;

src/p2p/p2p_build.c
234:     nlen = p2p->cfg->dev_name ? os_strlen(p2p->cfg->dev_name) : 0;
237:     wpabuf_put_data(buf, p2p->cfg->dev_name, nlen);
794:     if (p2p_add_wps_string(buf, ATTR_DEV_NAME, p2p->cfg->dev_name)

src/wps/wps_attr_parse.c
494:     attr->dev_name = pos;

src/wps/wps_enrollee.c
1063:         attr->dev_name, attr->dev_name_len);
1080:     m2d->dev_name = attr->dev_name;

src/wps/wps_er.c
50:     ev->dev_name = sta->dev_name;
83:     os_free(sta->dev_name);
821:     if (attr->dev_name) {
822:         os_free(sta->dev_name);
823:         sta->dev_name = dup_binstr(attr->dev_name, attr->dev_name_len);

src/wps/wps_dev_attr.c
388:     wps_process_dev_name(dev, attr->dev_name, attr->dev_name_len)
```

אחרי חיפוש בשורות השונות של תוצאת החיפוש, מצאנו את השורה המסומנת בחץ. זו השורה שבה מועתקים הביטים מתוך הבאפר אל המשתנים של התוכנה. מעקב אחרי שרשרת הקריאות הבאה מגלה כי מדובר בהעתקת ביטים מהזיכרון ללא בדיקה על התוכן אלא רק בדיקה של המצביע לתוכן (שאינו NULL):

`src/wps/wps_dev_attr.c/wps_process_dev_name`
`src/wps/wps_dev_attr.c/wps_process_device_attrs`
`src/wps/wps_er.c/wps_er_add_sta_data`

חיפשו בקוד את sta->dev_name:

```
src/wps/wps_er.c
50:   ev->dev_name = sta->dev_name;
83:   os_free(sta->dev_name);
822:  os_free(sta->dev_name);
823:  sta->dev_name = dup_binstr(attr->dev_name, attr->dev_name_len);
```

שורה 50 נמצאת בקישור הבא: src/wps/wps_er.c/wps_er_sta_event

המשתנה ev מצביע למבנה data.enrollee. מבנה זה נשלח בסוף הפונקציה אל

wps->event_cb(wps->cb_ctx, event, &data);, פונקציה העוסקת בפסיקות, בפרט גם אלו שמתרחשות

עבור לחיצת כפתור ה-WPS, זיהינו זאת כנקודת תורפה פוטנציאלית נוספת.

ATTR_ENCR_SETTINGS:

תוצאות החיפוש הראשוני:

```
src/wps/wps_attr_parse.c
514:   attr->encr_settings = pos;

src/wps/wps_enrollee.c
1010:  decrypted = wps_decrypt_encr_settings(wps, attr->encr_settings,
1121:  decrypted = wps_decrypt_encr_settings(wps, attr->encr_settings,
1174:  decrypted = wps_decrypt_encr_settings(wps, attr->encr_settings,
1244:  decrypted = wps_decrypt_encr_settings(wps, attr->encr_settings,

src/wps/wps_registrar.c
2845:  decrypted = wps_decrypt_encr_settings(wps, attr->encr_settings,
2996:  decrypted = wps_decrypt_encr_settings(wps, attr->encr_settings,
```

לפני שורה 1010 לא נעשות בדיקות נוספות, אך בפיענוח הבאפר המוחזר מהפונקציה מצאנו כי נעשות

בדיקות על המפתח:

```

struct wpabuf * wps_decrypt_encr_settings(struct wps_data *wps, const u8 *encr,
                                         size_t encr_len)
{
    struct wpabuf *decrypted;
    const size_t block_size = 16;
    size_t i;
    u8 pad;
    const u8 *pos;

    /* AES-128-CBC */
    if (encr == NULL || encr_len < 2 * block_size || encr_len % block_size)
    {
        wpa_printf(MSG_DEBUG, "WPS: No Encrypted Settings received");
        return NULL;
    }

    decrypted = wpabuf_alloc(encr_len - block_size);
    if (decrypted == NULL)
        return NULL;

    wpa_hexdump(MSG_MSGDUMP, "WPS: Encrypted Settings", encr, encr_len);
    wpabuf_put_data(decrypted, encr + block_size, encr_len - block_size);
    if (aes_128_cbc_decrypt(wps->keywrapkey, encr, wpabuf_mhead(decrypted),
                           wpabuf_len(decrypted))) {
        wpabuf_clear_free(decrypted);
        return NULL;
    }

    wpa_hexdump_buf_key(MSG_MSGDUMP, "WPS: Decrypted Encrypted Settings",
                        decrypted);

    pos = wpabuf_head_u8(decrypted) + wpabuf_len(decrypted) - 1;
    pad = *pos;
    if (pad > wpabuf_len(decrypted)) {
        wpa_printf(MSG_DEBUG, "WPS: Invalid PKCS#5 v2.0 pad value");
        wpabuf_clear_free(decrypted);
        return NULL;
    }
    for (i = 0; i < pad; i++) {
        if (*pos-- != pad) {
            wpa_printf(MSG_DEBUG, "WPS: Invalid PKCS#5 v2.0 pad "
                        "string");
            wpabuf_clear_free(decrypted);
            return NULL;
        }
    }
    decrypted->used -= pad;

    return decrypted;
}

```

למעשה, מצאנו כי כל ארבעת התוצאות הראשונות נמצאות בפונקציות:

`src/wps/wps_enrollee.c/wps_process_m[2,4,6,8]`

ראינו פונקציות אלה בתחילת המחקר. פונקציות אלו נקראות מתוך הפונקציה הראשית. בפונקציה הראשית

ראינו כי ה-ATTR השונים מפורסרים ולאחר מכן נשלחים לפונקציות שונות ב-switch-case הממייין לפי

סוג ההודעה בפרוטוקול.

את שתי התוצאות הבאות מצאנו בפונקציות:

src/wps/wps_registrar.c/wps_process_m[5,7]

פונקציות אלו שייכות להודעות המתקבלות אצל מכשיר הקצה, בעוד שמיקדנו את המחקר בהודעות המתקבלות אצל נקודת גישה. לכן, לא המשכנו לחקור הודעות, אך נשים לב שפונקציות אלה עושות שימוש באותה הפונקציה כמו הפונקציות של נקודת הגישה.

ATTR_CRED:

תוצאות החיפוש הראשוני:

```
src/wps/wps_common.c
445:         wpabuf_set(&msg, attr->cred[i], attr->cred_len[i]);

src/wps/wps_attr_parse.c
524:         attr->cred[attr->num_cred] = pos;
```

התוצאה היחידה הובילה לפונקציה הבאה:

```
int wps_oob_use_cred(struct wps_context *wps, struct wps_parse_attr *attr)
{
    struct wpabuf msg;
    size_t i;

    for (i = 0; i < attr->num_cred; i++) {
        struct wps_credential local_cred;
        struct wps_parse_attr cattr;

        os_memset(&local_cred, 0, sizeof(local_cred));
        wpabuf_set(&msg, attr->cred[i], attr->cred_len[i]);
        if (wps_parse_msg(&msg, &cattr) < 0 ||
            → wps_process_cred(&cattr, &local_cred)) {
                wpa_printf(MSG_ERROR, "WPS: Failed to parse OOB "
                    "credential");
                return -1;
            }
        wps->cred_cb(wps->cb_ctx, &local_cred);
    }

    return 0;
}
```

בפונקציה לעיל מופיעות שתי בדיקות, הראשונה מביניהם מחזירה אותנו לפונקציה ממנה התחלנו את המחקר, ואת הבדיקה השנייה, המסומנת בחץ, נחקור כעת:

src/wps/wps_attr_process.c/wps_process_cred

```

int wps_process_cred(struct wps_parse_attr *attr,
                    struct wps_credential *cred)
{
    wpa_printf(MSG_DEBUG, "WPS: Process Credential");

    /* TODO: support multiple Network Keys */
    if (wps_process_cred_network_idx(cred, attr->network_idx) ||
        wps_process_cred_ssid(cred, attr->ssid, attr->ssid_len) ||
        wps_process_cred_auth_type(cred, attr->auth_type) ||
        wps_process_cred_encr_type(cred, attr->encr_type) ||
        wps_process_cred_network_key_idx(cred, attr->network_key_idx) ||
        wps_process_cred_network_key(cred, attr->network_key,
                                     attr->network_key_len) ||
        wps_process_cred_mac_addr(cred, attr->mac_addr))
        return -1;

    return wps_workaround_cred_key(cred);
}

```

ניתן לראות שקיימת סדרת בדיקות לכאורה מדוקדקת, אך כשבדקנו לעומק מצאנו כי בחמש הפונקציות הבאות לא מתבצעות בדיקות מעבר לקיומו של מצביע חוקי (שאינו NULL):

```

src/wps/wps_attr_process.c/wps_process_cred_network_idx
src/wps/wps_attr_process.c/wps_process_cred_auth_type
src/wps/wps_attr_process.c/wps_process_cred_encr_type
src/wps/wps_attr_process.c/wps_process_cred_network_key_idx
src/wps/wps_attr_process.c/wps_process_cred_mac_addr

```

בפונקציה הבאה מתבצעת בדיקה תאימות בין האורך שדווח לבין זה שנשלח בפועל:

```

src/wps/wps_attr_process.c/wps_process_cred_ssid

```

בפונקציה הבאה מצאנו נקודת תורפה קצת יותר מעניינת:

```

static int wps_process_cred_network_key(struct wps_credential *cred,
                                       const u8 *key, size_t key_len)
{
    if (key == NULL) {
        wpa_printf(MSG_DEBUG, "WPS: Credential did not include "
                  "Network Key");
        if (cred->auth_type == WPS_AUTH_OPEN &&
            cred->encr_type == WPS_ENCR_NONE) {
            wpa_printf(MSG_DEBUG, "WPS: Workaround - Allow "
                      "missing mandatory Network Key attribute "
                      "for open network");
            return 0;
        }
        return -1;
    }

    wpa_hexdump_key(MSG_DEBUG, "WPS: Network Key", key, key_len);
    if (key_len <= sizeof(cred->key)) {
        os_memcpy(cred->key, key, key_len);
        cred->key_len = key_len;
    }

    return 0;
}

```

הפונקציה הנ"ל מאפשרת לדלג על הצורך ב-key network אם נשלחים ערכי ה-cred 'הנכונים' המופיעים בקובץ:

src/wps/wps_defs.h:

```
/* Authentication Type Flags */
#define WPS_AUTH_OPEN 0x0001
#define WPS_AUTH_WPAPSK 0x0002
#define WPS_AUTH_SHARED 0x0004 /* deprecated */
#define WPS_AUTH_WPA 0x0008
#define WPS_AUTH_WPA2 0x0010
#define WPS_AUTH_WPA2PSK 0x0020
#define WPS_AUTH_TYPES (WPS_AUTH_OPEN | WPS_AUTH_WPAPSK | WPS_AUTH_SHARED | \
    WPS_AUTH_WPA | WPS_AUTH_WPA2 | WPS_AUTH_WPA2PSK)
```

ATTR_NETWORK_KEY:

החיפוש על כל ההופעות של ה-ATTR היה ארוך מאוד, לכן בחרנו לסנן ולחפש את המקומות שבו האלמנט מועתק:

```
src/ap/wps_hostapd.c
527:     os_memcpy(hapd->wps->network_key, cred->key, cred->key_len);
1241:     os_memcpy(wps->network_key, conf->ssid.wep.key[0],
1491:     os_memcpy(wps->network_key, conf->ssid.wep.key[0],

src/wps/wps_registrar.c
1760:     os_memcpy(wps->cred.key, wps->wps->network_key,
```

התוצאה משורה 527 מופיע כחלק מהפונקציה:

src/ap/wps_hostapd.c/[hapd_wps_cred_cb](#)

שנקראת מתוך שרשרת הקריאות:

src/ap/wps_hostapd.c/[hostapd_wps_cred_cb](#)

src/ap/wps_hostapd.c/[hostapd_init_wps](#)

הפונקציה הראשונה בשרשרת היא אותה אחת שבה מופיעה תוצאת החיפוש הבאה (שורה 1241) ולכן עברנו לחקור אותה:

tree/src/ap/wps_hostapd.c/[hostapd_init_wps](#)

חקרנו את הפונקציה הזו קודם, עם **ATTR_MANUFACTURER**. בפונקציה עצמה אין בדיקות על הערכים המועתקים, וגם לא בשרשרת המובילה לקריאה לפונקציה הזו.

התוצאה בשורה 1491 מגיעה מהשרשרת:

src/ap/wps_hostapd.c/[hostapd_update_wps](#)

src/ap/hostapd.c/[hostapd_reload_bss](#)

שנקראת מתוך אחת משתי הפונקציות הבאות, כאשר הראשונה מבינה קוראת לה פעמיים:

src/ap/hostapd.c/[hostapd_reload_config](#)

src/ap/hostapd.c/[hostapd_reload_iface](#)

אמנם אין בפונקציות אלו בדיקות, אך הן פונקציות שנועדו לשחזר מידע, כך שאם בדרך כלשהיא הצלחנו להזריק לכאן ביטים - הם ישוחזרו ללא כל שינוי. זוהי עוד נקודת תורפה אפשרית, המהווה פתח להזרקת קוד שיקרא בכל מקום שבו נדרש ה-`network key`.

סיכום נקודות התורפה שזהינו:

- דיווח על `len` אחרון בצורה אסטרטגית שתגרום ללולאה ב-`wps_parse_msg` לסיים מוקדם או לרוץ לנצח
- ריפוד באפסים של שדות ה-`len` וה-`type` על מנת להזיז את `pos` לעמדה שאינה כפולה של זוגות בטים
- [ATTR_MODEL_NUMBER](#), [ATTR_MODEL_NAME](#), [ATTR_MANUFACTURER](#), [ATTR_SERIAL_NUMBER](#) נשמרים במבנה נתונים הנמצא בשימוש בכל פעם שנשלחת בקשת HTTP חוקית
- [ATTR_DEV_NAME](#) מועבר ללא בדיקות אל אזור בקוד הקשור ללחיצת הכפתור של WPS
- רוב האלמנטים במערך ה-`credentials` המועבר ב-[ATTR_CRED](#), אינם עוברים בדיקות כלל לפני העתקת התוכן, מלבד בדיקת תאימות אורך על `SSID`. בבדיקת ה-`credentials` של אלמנט ה-`network key` מצאנו ערכים מסוימים המוגדרים בקוד, שכשאר הם מועברים ניתן להימנע מהצורך להשתמש ב-`network key`
- [ATTR_NETWORK_KEY](#) אינו עובר בדיקות קפדניות

Fuzzer

בשלב השלישי, בחרנו להתמקד בהודעות ה-beacon וה-probe request ולבחון את נקודות התורפה האפשרויות שזיהינו בשלב השני באמצעות fuzzer.

בנינו fuzzer השולח את שתי ההודעות שציינו, ומוסיף לכל אחת מהן information element של WPS.

ה-information element בנוי באופן הבא:

```
wps = Dot11Elt(ID=221, info=(b'\x00\x50\xf2\x04' + b'\x10\x11' + b'\x00\x06' + b'\x48\x65\x64\x77\x69\x67'))
```

כאשר (משמאל לימין) החלק הראשון מסמן שמדובר בשדה Microsoft Corp. WPS, החלק השני מסמן את סוג ה-IE (במקרה ה"ל Device Name), החלק השלישי מסמן את אורך ה-IE (במקרה ה"ל האורך הינו 6) והחלק האחרון מכיל את תוכן ה-IE (במקרה ה"ל 'Hedwig').

ניתן לשנות את החלק השני לציין כל אחד מה-wps attributes לפי המספרים הבאים:

```
wps_attributes = {
    0x104A : {'name' : 'Version', 'type' : 'hex'},
    0x1044 : {'name' : 'WPS State', 'type' : 'hex'},
    0x1057 : {'name' : 'AP Setup Locked', 'type' : 'hex'},
    0x1041 : {'name' : 'Selected Registrar', 'type' : 'hex'},
    0x1012 : {'name' : 'Device Password ID', 'type' : 'hex'},
    0x1053 : {'name' : 'Selected Registrar Config Methods', 'type' : 'hex'},
    0x1038 : {'name' : 'Response Type', 'type' : 'hex'},
    0x1047 : {'name' : 'UUID-E', 'type' : 'hex'},
    0x1021 : {'name' : 'Manufacturer', 'type' : 'str'},
    0x1023 : {'name' : 'Model Name', 'type' : 'str'},
    0x1024 : {'name' : 'Model Number', 'type' : 'str'},
    0x1042 : {'name' : 'Serial Number', 'type' : 'str'},
    0x1054 : {'name' : 'Primary Device Type', 'type' : 'hex'},
    0x1011 : {'name' : 'Device Name', 'type' : 'str'},
    0x1008 : {'name' : 'Config Methods', 'type' : 'hex'},
    0x103C : {'name' : 'RF Bands', 'type' : 'hex'},
    0x1045 : {'name' : 'SSID', 'type' : 'str'},
    0x102D : {'name' : 'OS Version', 'type' : 'str'}
```

קישור: <https://github.com/devttys0/wps/blob/master/wpstools/wpscan.py#L12>

נפרט את הבדיקות שביצענו לפי סוג ההודעה, נציין שבכל בדיקה, שלחנו מספר רב של הודעות כדי לוודא שהתוצאה שהתקבלה לא התקבלה במקרה.

בהודעת ה-beacon ביצענו את הבדיקות הבאות:

1. כתבנו בשדה ה-SSID של WPS_IE SSID שונה ('Hedwig') מהאחד שנכתב בשדה ה-SSID של ה-frame ('mischief managed').
בניתוח ההודעה שנקלטה ב-Wireshark לא הופיעה הודעת שגיאה או אזהרה, וכך גם במכשיר ה-iphone, בו הרשת הופיע עם ה-SSID 'mischief managed'.

```
▶ IEEE 802.11 Beacon frame, Flags: .....
▼ IEEE 802.11 Wireless Management
  ▶ Fixed parameters (12 bytes)
  ▼ Tagged parameters (56 bytes)
    ▶ Tag: SSID parameter set: mischief managed
    ▶ Tag: RSN Information
    ▼ Tag: Vendor Specific: Microsoft Corp.: WPS
      Tag Number: Vendor Specific (221)
      Tag length: 14
      OUI: 00:50:f2 (Microsoft Corp.)
      Vendor Specific OUI Type: 4
      Type: WPS (0x04)
    ▼ SSID: Hedwig
      Data Element Type: SSID (0x1045)
      Data Element Length: 6
      SSID: Hedwig
```

על סמך הממצאים האלה, הסקנו שאין השפעה לשינוי זה, ייתכן שבדיקות מקיפות יותר על מכשירים נוספים היו מראים תוצאות שונות.

2. ציינו בשדה המציין את אורך ה-SSID ב-WPS_IE אורך קצר יותר (1) מהאורך האמיתי של ה-SSID שנשלח (6).

בניתוח ההודעה שנקלטה ב-Wireshark הופיעה הודעה שגיאה (Malformed Packet). בנוסף הופיע בתור ה-SSID ב-WPS רק התו הראשון של ה-SSID. כלומר, ה-SSID נקלט על פי האורך המופיע ב-IE ולא על פי האורך האמיתי של ה-SSID.
עם זאת, במכשיר ה-iphone הופיעה הרשת 'Hedwig' כרשת לגיטימית להתחבר אליה.
(נציין שבבדיקה זו, ה-SSID ב-frame תואם את ה-SSID ב-WPS_IE ('Hedwig').)

- ▶ IEEE 802.11 Beacon frame, Flags:
- ▼ IEEE 802.11 Wireless Management
 - ▶ Fixed parameters (12 bytes)
 - ▼ Tagged parameters (46 bytes)
 - ▶ Tag: SSID parameter set: Hedwig
 - ▶ Tag: RSN Information
 - ▼ Tag: Vendor Specific: Microsoft Corp.: WPS
 - Tag Number: Vendor Specific (221)
 - Tag length: 14
 - OUI: 00:50:f2 (Microsoft Corp.)
 - Vendor Specific OUI Type: 4
 - Type: WPS (0x04)
 - ▼ SSID: H
 - Data Element Type: SSID (0x1045)
 - Data Element Length: 1
 - SSID: H
- ▼ [Malformed Packet: IEEE 802.11]
 - ▼ [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
 - [Malformed Packet (Exception occurred)]
 - [Severity level: Error]
 - [Group: Malformed]

על סמך הממצאים האלה, ניתן להסיק שהמכשירים העומדים לרשותנו שקלטו את הודעת ה-beacon לא מבצעים בדיקות מעמיקות על שדות אלה בשלב זה, ולכן לא התריעו על בעיה ברשת. אך ייתכן, שהתראה כזאת כן הופיעה, אך לא ניתן לזהות אותה מהתבוננות בלבד ברשתות הזמינות למכשירים אלו.

3. ציינו בשדה המציין את אורך ה-SSID ב-WPS_IE אורך יותר (8) מהאורך האמיתי של ה-SSID שנשלח (6).

בניתוח ההודעה שנקלטה ב-Wireshark הופיעה הודעה שגיאה (Malformed Packet). בנוסף, לא הופיע שדה ה-SSID ב-WPS כלל!

במכשיר ה-iphone הופיעה הרשת 'Hedwig' כרשת לגיטימית להתחבר אליה.

- ▶ IEEE 802.11 Beacon frame, Flags:
- ▼ IEEE 802.11 Wireless Management
 - ▶ Fixed parameters (12 bytes)
 - ▼ Tagged parameters (46 bytes)
 - ▶ Tag: SSID parameter set: Hedwig
 - ▶ Tag: RSN Information
 - ▼ Tag: Vendor Specific: Microsoft Corp.: WPS
 - Tag Number: Vendor Specific (221)
 - Tag length: 14
 - OUI: 00:50:f2 (Microsoft Corp.)
 - Vendor Specific OUI Type: 4
 - Type: WPS (0x04)
 - ▼ [Malformed Packet: IEEE 802.11]
 - ▼ [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
 - [Malformed Packet (Exception occurred)]
 - [Severity level: Error]
 - [Group: Malformed]

על סמך הממצאים האלה, ניתן לראות ששינוי זה אינו מאפשר לגשת לזיכרון שאינו שייך לנו, ואף מבטל שדה זה. ניתן להסיק, שקיימות בדיקות מקיפות על שינויים מסוג זה. עם זאת, מכיוון שהרשת הופיעה כרשת לגיטימית ברשימת הרשתות הזמינות במכשיר ה-iphone, ייתכן שבשלב זה, שגיאה זו אינה מהווה שגיאה מספיק חמורה כדי להגדיר את הרשת כלא חוקית עפ"י המכשיר.

אך עם זאת, ייתכן שהופעת הרשת ברשימת הרשתות הזמינות אינה מהווה מדד מספק כדי לדעת האם המכשיר מזהה שינוי זו או לא.

4. כתבנו בשדה ה- SSID ב- WPS_IE SSID המכיל את התו Null ולאחריו תווים מיוחדים באופן הבא:
'Hedwig\n%p%s%n'
בניתוח ההודעה שנקלטה ב- Wireshark הופיעה הודעה התראה (Trailing stray characters).
אך השם המופיע ב- SSID ב- WPS_IE הופיע ללא התווים המיוחדים, כלומר הופיע השם 'Hedwig' בלבד.

במכשיר ה- Iphone הופיעה הרשת 'Hedwig' כרשת לגיטימית להתחבר אליה.

```
▶ IEEE 802.11 Beacon frame, Flags: .....
- IEEE 802.11 Wireless Management
  ▶ Fixed parameters (12 bytes)
  - Tagged parameters (53 bytes)
    ▶ Tag: SSID parameter set: Hedwig
    ▶ Tag: RSN Information
    - Tag: Vendor Specific: Microsoft Corp.: WPS
      Tag Number: Vendor Specific (221)
      Tag length: 21
      OUI: 00:50:f2 (Microsoft Corp.)
      Vendor Specific OUI Type: 4
      Type: WPS (0x04)
    - SSID: Hedwig
      Data Element Type: SSID (0x1045)
      Data Element Length: 13
    - SSID: Hedwig
      - [Expert Info (Warning/Undecoded): Trailing stray characters]
        [Trailing stray characters]
        [Severity level: Warning]
        [Group: Undecoded]
```

על סמך הממצאים האלה, ניתן לראות שניתן להסתיר תווים מסוימים ב- SSID המופיע ב- WPS. ניתן להסיק מכך, שככל הנראה הבדיקות המתבצעות על ה- SSID אינן מונעות הוספת תווים מיוחדים לאחר תו Null ולא מגדירות זאת כשגיאה. ייתכן וניתן לנצל זאת, בין אם בהתחזות לרשת לגיטימית ובין אם ע"י הוספת קוד ל- SSID. עם זאת, ייתכן ומתבצעות במכשירים בדיקות יותר מקיפות בהודעות המשך, אשר מונעות זאת.

5. ציינו בשדה המציין את אורך ה- Device Name ב- WPS_IE אורך קצר יותר (1) מהאורך האמיתי של ה- Device Name שנשלח (6).
בניתוח ההודעה שנקלטה ב- Wireshark הופיעה הודעה שגיאה (Malformed Packet). בנוסף הופיע בתור ה- Device Name ב- WPS רק התו הראשון של ה- Device Name. כלומר, ה- Device Name נקלט על פי האורך המופיע ב- IE ולא על פי האורך האמיתי של ה- Device Name.

- ▶ IEEE 802.11 Beacon frame, Flags:
 - ▼ IEEE 802.11 Wireless Management
 - ▶ Fixed parameters (12 bytes)
 - ▼ Tagged parameters (56 bytes)
 - ▶ Tag: SSID parameter set: mischief managed
 - ▶ Tag: RSN Information
 - ▼ Tag: Vendor Specific: Microsoft Corp.: WPS
 - Tag Number: Vendor Specific (221)
 - Tag length: 14
 - OUI: 00:50:f2 (Microsoft Corp.)
 - Vendor Specific OUI Type: 4
 - Type: WPS (0x04)
 - ▼ Device Name: **H**
 - Data Element Type: Device Name (0x1011)
 - Data Element Length: 1
 - Device Name: H
- ▼ [Malformed Packet: IEEE 802.11]
 - ▼ [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
 - [Malformed Packet (Exception occurred)]
 - [Severity level: Error]
 - [Group: Malformed]

במכשיר ה- Iphone הופיעה הרשת 'mischief managed' כרשת לגיטימית להתחבר אליה.

6. ציינו בשדה המציין את אורך ה- Device Name ב- WPS_IE אורך יותר (8) מהאורך האמיתי של ה- Device Name שנשלח (6).

בניתוח ההודעה שנקלטה ב- Wireshark הופיעה הודעה שגיאה (Malformed Packet). בנוסף, לא הופיע שדה ה- Device Name ב- WPS כלל! במכשיר ה- Iphone הופיעה הרשת 'mischief managed' כרשת לגיטימית להתחבר אליה.

- ▶ IEEE 802.11 Beacon frame, Flags:
- ▼ IEEE 802.11 Wireless Management
 - ▶ Fixed parameters (12 bytes)
 - ▼ Tagged parameters (56 bytes)
 - ▶ Tag: SSID parameter set: mischief managed
 - ▶ Tag: RSN Information
 - ▼ Tag: Vendor Specific: Microsoft Corp.: WPS
 - Tag Number: Vendor Specific (221)
 - Tag length: 14
 - OUI: 00:50:f2 (Microsoft Corp.)
 - Vendor Specific OUI Type: 4
 - Type: WPS (0x04)
 - ▼ [Malformed Packet: IEEE 802.11]
 - ▼ [Expert Info (Error/Malformed): Malformed Packet (Exception occurred)]
 - [Malformed Packet (Exception occurred)]
 - [Severity level: Error]
 - [Group: Malformed]

על סמך הממצאים האלה, ניתן לראות ששינוי זה אינו מאפשר לגשת לזיכרון שאינו שייך לנו, ואף מבטל שדה זה.

7. כתבנו בשדה ה- Device Name ב- WPS_IE המכיל את התו Null ולאחריו תווים מיוחדים באופן הבא: 'Hedwig\n%k%s%n' וציינו בשדה האורך את האורך האמיתי של ה- Device Name.

בניתוח ההודעה שנקלטה ב-Wireshark הופיעה הודעה התראה (Trailing stray characters).
במכשיר ה-Iphone הופיעה הרשת 'mischief managed' כרשת לגיטימית להתחבר אליה.

- ▶ IEEE 802.11 Beacon frame, Flags:
- ▼ IEEE 802.11 Wireless Management
 - ▶ Fixed parameters (12 bytes)
 - ▼ Tagged parameters (63 bytes)
 - ▶ Tag: SSID parameter set: mischief managed
 - ▶ Tag: RSN Information
 - ▼ Tag: Vendor Specific: Microsoft Corp.: WPS
 - Tag Number: Vendor Specific (221)
 - Tag length: 21
 - OUI: 00:50:f2 (Microsoft Corp.)
 - Vendor Specific OUI Type: 4
 - Type: WPS (0x04)
 - ▼ Device Name: Hedwig
 - Data Element Type: Device Name (0x1011)
 - Data Element Length: 13
 - ▼ Device Name: Hedwig
 - ▼ [Expert Info (Warning/Undecoded): Trailing stray characters]
 - [Trailing stray characters]
 - [Severity level: Warning]
 - [Group: Undecoded]

על סמך הממצאים אלה, ניתן להסיק שככל הנראה הבדיקות המתבצעות על ה-Device Name אינן מונעות הוספת תווים מיוחדים, ואף ייתכן אינן מונעות הזרקת קוד.

לסיכום, על סמך תוצאות הבדיקות שביצענו, בסבירות גבוהה מתבצעות בדיקות על אורך ה-attribute שנכתב אל מול האורך שנכתב ב-IE של attribute זה. בדיקות אלה מונעות גישה לקוד שאין לנו הרשאות גישה אליו במקרה שבו האורך שצוין ארוך יותר מהאורך האמיתי. התוצאה המבטיחה ביותר שקיבלנו הינה של הבדיקות ה-4 וה-7 שבהן הוספנו תווים מיוחדים שלא הופיעו בשם שהוצג בניתוח ההודעה ב-Wireshark. ייתכן וניתן לנצל מצב זה בשביל הוספת קוד או התחזות לרשת אחרת. עם זאת, חשוב לציין שבכל הבדיקות, במכשיר ה-Iphone הופיעה הרשת כרשת לגיטימית להתחבר אליה. לכן, הגענו למסקנה שהופעת הרשת ברשימת הרשתות הזמינות שהמכשיר מציג, אינה מהווה מדד מספק להשפעת הבדיקה על המכשיר, לכן יש למצוא דרך אחרת אשר תאפשר לבדוק את השפעת הבדיקות המתבצעות.

בהודעת ה-probe request ביצענו את הבדיקות הבאות:

נציין שבהודעות ה-probe response הנשלחת ע"י נקודת הגישה הנמצאת ברשותנו, לא מופיע שדה ה-SSID ב-WPS_IE, אך מופיע שדה ה-Device Name ('RalinkAPS'), ולכן בחרנו להתמקד בשדה זה בבדיקות שביצענו.

1. כתבנו בשדה ה- Device Name של WPS_IE של נקודת הגישה אליה אנו מבקשים להתחבר Device Name שונה ('Hedwig'), מהאחד שנכתב בשדה ה- Device Name בהודעות של נקודת הגישה ('RalinkAPS').

בניתוח הודעת ה- probe response שקיבלנו מנקודת הגישה אל הכתובת ה- MAC ממנה שלחנו את הודעת ה- probe request לא הופיע כל שינוי מההודעה הרגילה, ובין השאר, בשדה ה- Device Name הופיע השם התקני ('RalinkAPS').

2. ציינו בשדה המציין את אורך ה- Device Name ב- WPS_IE אורך יותר (13) מהאורך האמיתי של ה- Device Name שנשלח (9), בעוד שכתבנו את ה- Device Name התקני של נקודת הגישה ('RalinkAPS').

בניתוח הודעת ה- probe request ב- Wireshark הופיעה הודעת שגיאה (Malformed Packet), ולא הופיע שדה ה- Device Name ב- WPS, בדומה לשגיאה שהופיעה בניתוח הודעת ה- beacon בבדיקה דומה.

בניתוח הודעת ה- probe response שקיבלנו מנקודת הגישה אל הכתובת ה- MAC ממנה שלחנו את הודעת ה- probe request לא הופיע כל שינוי.

על סמך הממצאים האלה, ניתן להסיק שלמרות שהופיעה שגיאה בניתוח הודעת ה- probe request, עדיין נשלחת הודעת probe response תקנית. כלומר, שגיאה זו ככל הנראה לא משפיעה על התנהגות נקודת הגישה.

3. ציינו בשדה המציין את אורך ה- Device Name ב- WPS_IE אורך קצר יותר (1) מהאורך האמיתי של ה- Device Name שנשלח (9), בעוד שכתבנו את ה- Device Name התקני של נקודת הגישה ('RalinkAPS').

בניתוח הודעת ה- probe request ב- Wireshark הופיעה הודעת שגיאה (Malformed Packet), ושדה ה- Device Name ב- WPS הכיל תו יחיד בלבד, בהתאם לאורך שציינו ב- WPS_IE. זאת, בדומה לתוצאות שקיבלנו בבדיקה דומה בהודעת ה- beacon.

בניתוח הודעת ה- probe response שקיבלנו מנקודת הגישה אל הכתובת ה- MAC ממנה שלחנו את הודעת ה- probe request לא הופיע כל שינוי.

על סמך הממצאים האלה, ניתן להסיק שלמרות שהופיעה שגיאה בניתוח הודעת ה- probe request, עדיין נשלחת הודעת probe response תקנית. כלומר, שגיאה זו ככל הנראה לא משפיעה על התנהגות נקודת הגישה.

לסיכום, על סמך תוצאות הבדיקות שביצענו, הגענו למסקנה שבסבירות גבוהה אין השפעה לשינוי בשדות בהודעת ה- probe request על השדות בהודעת ה- probe response המתקבלת בחזרה, ואף על התנהגות נקודת הגישה. יתרה מזאת, לפי התוצאות שקיבלנו, שגיאה בשדות ה- probe request אינה מונעת קבלת הודעת probe response. עם זאת, חשוב לציין, כי ייתכן ושינוי בשדות אחרים שלא בדקנו, או שינויים אחרים בשדות שבדקנו, יכולים להשפיע על נקודת הגישה.

לוח זמנים

לוח זמנים הנקבע בדו"ח הפתיחה

שלב בפרויקט	טווח תאריכים
היכרות מעמיקה עם תקן WPS	01/11-13/11
בחירת הנקודות לתקיפה	07/11-13/11
תכנון ה-fuzzer והקלטים	14/11-20/11
תכנון שיטה לבדיקת תוצאות	21/11-27/11
בניית ה-fuzzer	21/11-18/12
הרצת ניסויים ובחינת התוצאות	19/12-15/01
פוסטר	16/01-19/01
תקופת מועדי א'	19/01-17/02
כתיבת הדו"ח הסופי	08/03-19/03
תקופת מועדי ב'	25/02-13/07
כתיבת מצגת סיכום	20/03-27/03

לוח הזמנים בפועל

כאשר כתבנו את דו"ח הפתיחה, טרם התקנו ולמדנו לעבוד עם הכלים הנדרשים לביצוע הפרויקט. שלב זה, התברר כשלב הבעייתי ביותר בפרויקט שמנע מאיתנו להתקדם בקצב המתוכנן מבחינת כתיבת ה-fuzzer והרצת הניסויים. לכן, רוב הפרויקט, עסקנו במחקר של הקוד הפתוח שמצאנו המממש את תקן ה-WPS וחיפוש אחר פרצות אבטחה פוטנציאליות בו. לאחר שטיפלנו בבעיות הטכניות שנתקלנו, התקדמו בקצב יחסית מהיר עד לסיום הפרויקט. נציין שבעקבות שינוי תקופת המבחנים בעקבות מגפת הקורונה והוספת מועדי ג', הגשת הפרויקט נדחתה ל- 13/04 ורוב העבודה הקשורה ב-fuzzer התבצעה בתקופה שבין מועדי ב' למועדי ג'.

סיכום והצעות לשיפור

המוטיבציה בביצוע הפרויקט הייתה מחקר וחשיפה של פרצות אבטחה שונות בתקן ה-WPS. במהלך הפרויקט התנסו במעבר על קוד פתוח המממש את התקן וחיפוש אחר פרצות אבטחה פוטנציאליות. בנוסף, התנסו בכתיבת fuzzer המבצע בדיקות המאפשרות לבחון היתכנות של פרצות אבטחה אלה בנקודת גישה ובמכשירי קצה אשר עמדו לרשותנו. ולבסוף, התנסו בשימוש בכלים מגוונים כגון Wireshark ו-Scapy המשמשים לבחינה וניתוח פרוטוקולי תקשורת שונים.

אמנם, לא הצלחנו לנצל את פרצות האבטחה הפוטנציאליות שבדקנו כדי להשיג את אחת המטרות שהצבנו לעצמנו בפרויקט: לגרום למכשיר הקצה או לנקודות גישה להתנהגות לא רצויה, במכשירים שעמדנו לרשותנו. אך אנחנו מעריכות שבדיקות מקיפות ומעמיקות יותר, בהתבסס על תוצאות מחקר הקוד, על מכשירים נוספים בהם התקן ממומש באופן דומה, עשוי להוביל למציאת פרצות אבטחה.