תרגיל בית 2

שמחים ונרגשים להתחיל את תרגיל הבית ה-2 התחברנו לאתר והתחלנו להיכנס למשתמשים השונים בעזרת הסיסמאות אותנו חילצנו בתרגיל הקודם.

ראינו כי לכל משתמש יש בתפריט "Files" ושם נמצאות כספות שונות. החלנו לחטט באתר והורדנו את כל הקבצים מהכספות. ראינו כי בכספת הציבורית מופיע pdf עם הסבר על מהלך התרגיל – עלינו לפתור את הכספות של כל אחד משלושת השחקנים (כאשר ה-archer הוזנח בתרגיל), ובעזרת מפתחות ופרמטרים שנקבל בפתיחת הכספות שלהם נוכל לגשת לקובץ decrypt.exe ולחלץ סיסמה לכספת המשותפת שבאתר.

הפשלנו שרוולים והתחלנו לעבוד.

giant

תחילה, מצאנו את פונק' ה-main כפי שלמדנו בסדנה, ובה מצאנו 3 פונק' להן ביצענו rename לשמות הבאים: first_part, second_part ו-third_part (מכאן והלאה נשתמש בשמות אלו).

first_part:

דבר ראשון, ראינו על-פי ה-strings שקיימת קריאה ל-scanf בחלק זה בתוך פונק' פנימית הנקראת מ-first_part. כדי להבין מה עלינו לסרוק, התחלנו לנתח את הפונקציות.

ב-first_part ראינו כי קודם כל מוגדר מערך עם 8 איברים, ואחרי הגדרתו הוא מועבר יחד עם אינדקס הסיום (7) ואינדקס ההתחלה (0) לפונק' הפנימית (יחד עם ארג' נוסף).

בחזרה מהפונק' הפנימית ראינו כי מתבצעת לולאה המוודאת כי המערך ממויין (בכל איטרציה נבדק האם האיבר ה-i קטן מהאיבר ה-i+1), אחרת יש exit בהמשך, ובנוסף מתבצעות הדפסות שונות (על-ידי פונק' ששינינו את שמה להיות printer_func שמשתמשת ב-putc ומדפיסה תו-תו ככמות התווים שהועברו בארג').

כאן הבנו שעלינו למיין את המערך, ואת זה יש לבצע באמצעות הפונק' הפנימית!

בפונק' הפנימית מתבצעת סריקת של 5 מספרים ב-hex של שתי ספרות. לאחר דיבוג, ראינו שהערכים שנסרקו מועתקים על קטע הקוד בהמשך הפונק' שמכיל רצף של פקודות nop (ההעתקה על-ידי VirtualProtect ו-memset).
בשאר הפונק' ראינו כי מתבצעת חלוקה של אינדקס הסיום של המערך ב-2, ואז קריאה רקורסיבית לאותה פונק' עם אותו מערך, כאשר אינדקס הסיום מוחלף בתוצאת החלוקה. לאחר מכן, מתבצעת הכנה של ארגומנטים לקראת קריאה לפונק', אך אף פונק' אינה נקראת.
הארגומנטים הם שוב המערך, מועבר האינדקס לאחר החלוקה ב-2 (+1), ואינגקס הסיום המקורי.
לאחר מכן מתבצעת קריאה לפונק' נוספת (שממבט חטוף בה ראינו כי היא מבצעת שינוי בסדר האיברים המערך).
לאחר נסיונות כשולים, וכאשר אנו יודעים כי אנו רוצים לבצע מיון של המערך, וכן בעזרת אינטואציה שאנו מחלקים פה את המערך ל-2 (ולכן האינדקס מחולק ל-2 ומועברים אינדקסי התחלה וסיום כפי שפירטנו) – הגענו למסקנה שכנראה יש פה מימוש כלשהו של merge sort ולכן מה שעלינו לסרוק (ושיחליף את פקודות ה-nop) הוא קריאה רקורסיבית לפונק' הפנימית עם חצי המערך השני!

בדקנו מהו ה-offset מפקודת ה-nop השישית (אליה יצביע EIP לאחר קריאת הבתים שנכניס) אל תחילת הפונק' ומצאנו:

$$0x401683 - 0x4017B9 = -0x136$$

ולאחר המרה למשלים ל-2 (ושימוש ב-32 ביטים): $FF\ FF\ FE\ CA$.
לכן המספרים אותם סרקנו הם קידוד הפקודה call וה-offset (ב-little endian), כלומר:

$$E8\ CA\ FE\ FF\ FF$$

second_part:

בפונק' זו תחילה מתבצעת קריאה ל-SetUnhandledExceptionFilter, שבקריאה קצרה ב-api של וינדוס גילינו כי היא מאפשרת להגדיר פונק' שתיקרא בעת חריגה.

עברנו על ה-handler וראינו כי תחילה הוא ניגש ל-ExceptionRecord של החריגה (struct של וינדוס) ומוודא את הדברים הבאים:

1) על-ידי גישה ל-ExceptionCode, מוודא כי החריגה שהתקבלה היא 0xC0000005, שמשמעותה היא "Access Violation Exception".

2) על-ידי גישה ל-ExceptionInformation[1], מוודא כי הכתובת אליה ניסינו לגשת היא 0, כלומר NULL (בדקנו ב-api וגילינו שעבור חריגה מסוג זה, מה שנשמר במערך הנ"ל באינדקס 1 הוא הכתובת אליה ניסינו לגשת).

אם הבדיקות הללו עברו, אזי יש הכנסה של הערך 1 למשתנה ב-data section (קראנו לו exception_flag) שינוי של איזור הזכרון בו קרתה החריגה, ועל-ידי memset הפקודות באיזור זה משתנות ל-0x90, כלומר ל-nop (כדי שלא תקרה חריגה שוב בעת הרצת הפקודה).

חזרנו לפונק' second_part וראינו כי הייצוג ב-ascii של "NLUL" מוכנס לאחד המשתנים המקומיים. לאחר מכן, נסרקים 4 תווים, ומתבצעת החלפה בסדר של שני התווים האמצעיים (באמצעות shl), כך שאם למשל נסרוק "NULL" נקבל "NLUL".
מתבצע חיסור בין הבתים שסרקנו לבין "NLUL", ומתבצעת הדפסה של המחרוזת הזאת שמתקבלת מהחיסור. לאחר ההדפסה ישנה בדיקה כי ה-exception_flag הינו 1, כלומר מוודאים כי נכנסנו ל-handler. לפיכך, אנו רוצים לוודא שניגשנו ל-NULL לפני הבדיקה הזאת – דרך פשוטה לעשות זאת היא לסרוק "NULL" ולקבל בחיסור 0 (כלומר NULL).

:third_part

ראינו כי נסרק מספר ב-hex, ואם המספר שנסרק הוא שלילי, מבצעים לו neg. נסמנו x0.
כמו כן, נסרקים 3 מספרים דצימליים, שנסמנם x1, x2 ו-x3.
לאחר מכן מתבצעות מספר בדיקות על המספרים האלו:

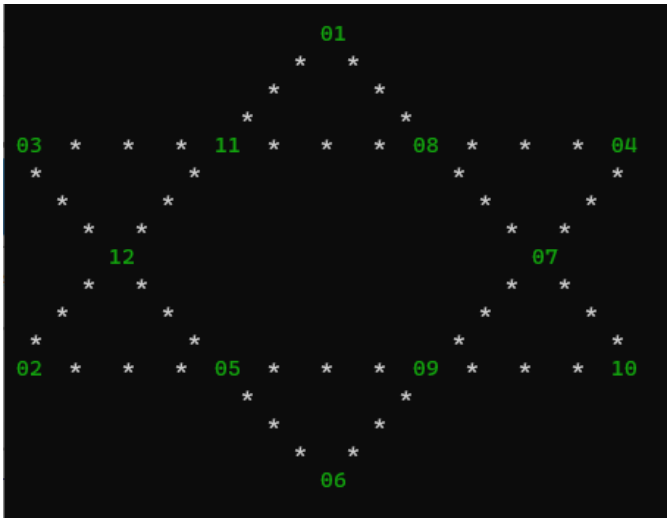
1) מוודאים ש-x0 הוא שלילי, למרות שלפני כן אם הכנסנו מספר שלילי, ביצענו לו neg. המסקנה היא שיש להכניס את המספר הנמוך ביותר הניתן לייצוג – 0x80000000, כיוון שבעת ביצוע neg לאחר הפיכה של הביטים וביצוע 1+ נקבל שוב את אותו מספר, וזהו המספר השלילי היחיד בו זה יקרה.
2) ה-lsb של x1 הוא 0xCB=203, ולכן בחרנו ב-x1=203.
3) x2 אמור להיות שלילי, אך בנוסף x2-1 אמור להיות אי-שלילי. זה כמובן לא אפשרי עם כל מספר שלילי "רגיל", אך שוב נוכל להשתמש במספר השלילי ביותר הניתן לייצוג ובאופן דומה נקבל לאחר ביצוע ה-1+ מספר אי-שלילי (בגלל ענייני ייצוג). בייצוג דצימלי מספר זה הוא 2147483648-.
4) מתבצע חישוב של $-43 \cdot (x_3)^2 - 7224 \cdot x_3 + 303407$, ואז בדיקה כי התוצאה שווה ל-1. כתבנו סקריפט פייתון שעובר על ערכים אפשריים בטווח $[-10000, 10000]$ וגילינו כי x3=-84 מקיים זאת.

wizard

גם כאן מצאנו את פונק' ה-main וביצענו rename לפונק' הנקראות מתוכו ל-first_part ו-second_part.

first_part:

הרצנו את הקובץ וראינו כי מתבצעת הדפסה של מגן דוד עם סימני שאלה בקודקודים.
לאחר ניתוח הקוד, ראינו כי מתבצעת בדיקה על האלכסונים של מגן הדוד (וכן על כל הקודקודים) כך שסכום כל אחד מהם יהיה שווה ל-26.
כיוון שהתעצלנו לפתור את החידה בעצמנו, חיפשנו באינטרנט פתרונות לחידה, ואכן מצאנו כאלו ב-math exchange. כמו כן, שמנו לב כי המספרים שנבחרים כקודקודים אינם אלו שאנו סורקים, אלא נלקחים מתוך מערך מוגדר. המספרים שאנו סורקים הם האינדקסים בהם המספרים הרצויים נמצאים במערך (ומשם הם נלקחים). אם כך, "תרגמנו" את פתרון החידה שמצאנו על האינדקסים הרצויים, וקיבלנו את הפתרון:



second_part:

ראינו כי תחילה נסרקת מחרוזת כלשהי על-ידינו, ולאחר מכן מתבצעות מניפולציות על אותה מחרוזת:

1) מכל אות במחרוזת מחסרים את הערך 65, כלומר האות "A".
2) מבצעים XOR של כל אות במחרוזת עם האות הבאה (באופן ציקלי).
3) מוסיפים לכל במחרוזת את הערך 65, כלומר האות "A".

לאחר המניפולציה מתבצעת השוואה של המחרוזת הנ"ל לבין מחרוזת קבועה שמוגדרת בתוכנית: "IDAEKGGEFGILKNLIJKCHMEABAKHLM".

כתבנו תוכנית c שמבצעת את הפעולות ההופכיות לפעולות אלו וכך גילינו מהי המחרוזת עלינו לסרוק:

```
int main()
{
    char str[29] = "IDAEKGGEFGILKNLIJKCHMEABAKHLM";
    int i = 0;

    printf("%s\n", str);
```

```
    for(int j=0; j<29; j++) {
        str[j] -= 'A';
    }

    for(int j=0; j<29; j++) {
        str[28 - j] ^= str[(28 - j + 1) % 29];
    }

    for(int j=0; j<29; j++) {
        str[j] += 'A';
    }

    printf("%s\n", str);

    return 0;
}
```

וקיבלנו את המחרוזת: IADDHNLNJMKCJDOFNEOMLHDDCCIPE.

לדאבוננו, גילינו כי השלב אינו נגמר.

בהמשך מתבצע שרשור של הכתובת של second_part אל המחרוזת שסרקנו (שנמצאת על המחסנית) על-ידי הפונק' sprintf. לאחר השרשור, מתבצעת בדיקה כי כתובת החזרה השמורה על המחסנית הינה second_part (מה שכמובן לא קורה במצב הרגיל, כיוון שהפונק' נקראה מ-main, ולכן כתובת החזרה היא ל-main).

אם כך, נרצה לדרוס את כתובת החזרה על המחסנית ולשנות אותה ל-second_part. למזלנו, בשרשור כבר נכתבת הכתובת של second_part על המחסנית (בהמשך לשרשרת שסרקנו), ולכן כל שנותר לנו לעשות הוא לדאוג שהכתיבה תדרוס את כתובת החזרה ששמורה במחסנית.

חזרנו אל האיזור בו מוגדרים המשתנים של הפונק' הנוכחית וגילינו כמה בתים יש בין המחרוזת שאנו סורקים לבין כתובת החזרה, והארכנו את המחרוזת שאנו סורקים כך שתמלא גם את האיזור הזה ולכן השרשור ידרוס את כתובת החזרה. לאחר ספירה מעמיקה, גילינו שיש 16 בתים, ולכן שרשרנו למחרוזת שמצאנו גם 16 תווי '0'.

goblin

כמיטב המסורת מצאנו את פונק' ה-main אך להפתעתנו לא היו שלבים שונים.

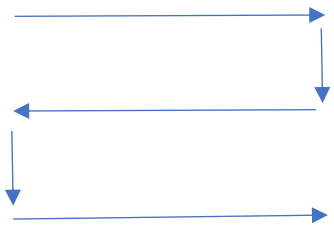
בהתחלה מוגדרים משתנים רבים. ומתבצעת קריאה לפונק' לה קראנו init_maze (לאחר זמן רב, כשהצלחנו להבין מה היא עושה). בפונק' זו מבצעים אתחולים רבים: המשתנים x ו-y שמועברים מ-main (כפוינטרים) מאותחלים ל-0, וכן המשתנה direction שגם הוא מועבר מ-main (כפוינטר) מאותחל ל-R. כמו כן, איזור מסוים בזכרון מאותחל ב-'X'-ים ו-'.'-ים, ומשבצות מסוימות באותו איזור זכרון מאותחלות למספרים כלשהם.
כמו כן מאותחלים שני משתנים בזכרון להם בדיעבד קראנו target_row=7 ו-target_col=0.

לאחר האתחול, מתחילה לולאה המוגבלת ל-7 איטרציות (בדיעבד – 7 צעדים לשחק).

אחר כך נקראת פונק' בה מתבצע scanf, ולכן קראנו לה בדיעבד choose_action. בפונק' זו נסרק char כלשהו מבין התווים המותרים – 'C', 'D' ו-'U' (בדיעבד – continue, down ו-up).
במידה ונסרק 'C', כלומר continue, נסרק תו נוסף מבין התווים המייצגים ספרות ('0'-'9'), ונשמרת הספרה שנבחרה כ-int (מוחסר ממנה '0').

לאחר מכן נקראות אחת מהפונק' שמבצעות את הפעולה שנבחרה כתלות באיזו פעולה זו:

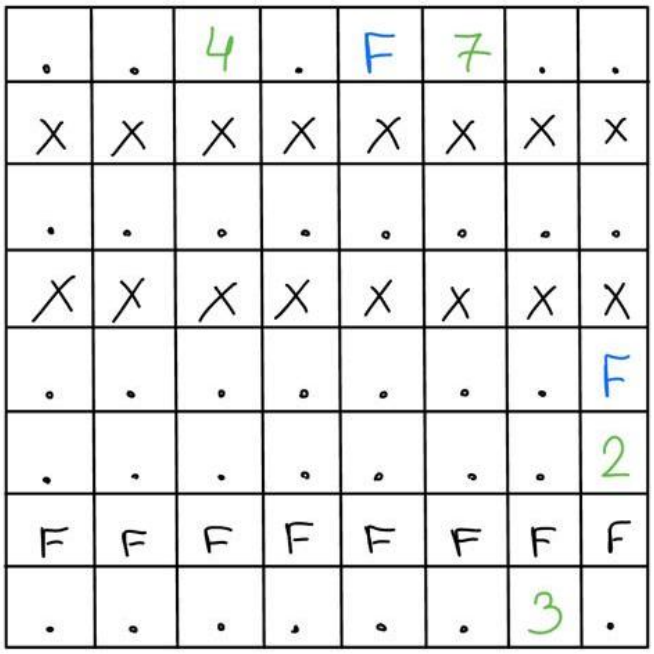
1) אם נבחר 'C', אזי נקראת פונק' שמזיזה את x ו-y בהתאם לכמות הצעדים שנבחרה, כאשר התזוזה היא ב-"זיגזג", כלומר באופן הבא:



כאשר בעצם x ו-y מייצגים את המיקום על הלוח בגודל 8X8.

2) אם נבחר 'U' או 'D', אזי בפונק' הנקראת מוודאים שאנו נמצאים על משבצת בעלת ערך מספרי (ולא 'X' או '.'). אנו "נוקפץ" למעלה/למטה (בהתאם לבחירה שלנו) על הלוח, כאשר הכמות המשבצות בהן נוקפץ הן כערך המספרי שמופיע במשבצת עליה אנו עומדים.

נצרף את הלוח שאותחל (כאשר כתובות הזכרון עולות בשורות התחתונות יותר, ולכן כדי לרדת בלוח המצורף בעצם יש לבחור ב-'U'):



ראינו כי תנאי יציאה מהלולאה (ונצחון במשחק) הוא כאשר x=target_row ו-y=target_col. לכן, על-פי חוקי המשחק אותם הבנו, עלינו למצוא מסלול המוביל אותנו למשבצת המטרה.

לאחר שמצאנו מסלול, אך בכל זאת נפסלנו, שמנו לב לתנאי נוסף – אם המיקום הנוכחי נמצא על F, אז מתבצע 1+ ל-counter המיועד לכך. לבסוף, נבדק גם כי ה-counter שווה ל-2.
כלומר, יש לדאוג כי במסלול אנו נעבור בשתי המשבצות בהן מופיע F. כל שנותר הוא למצוא מסלול שעונה על שני התנאים, ועשינו זאת.

decrypt

לאחר שפיצחנו את שלושת הכספות של המשתמשים השונים עברנו לקובץ ה-decrypt.

ניתחנו את הקובץ והבנו מהו סדר הפרמטרים הדרוש – הפרמטר הראשון מיועד לקבוע את מספר הפעמים שלולאה מסוימת תתבצע, ולכן הפרמטר הראשון הינו ה-rounds שקיבלנו.
הפרמטר השני משמש באותה לולאה לקידום של איטרטור ולכן הפרמטר המתאים הינו ה-delta.
הפרמטר הרביעי הוא ה-key, שכן נבדק שהוא באורך 16, אחרת מודפס שה-key לא באורך המתאים.
וכל שנותר הוא לשים את ה-hashed password כפרמטר השלישי.

בנסיון הראשון, הסיסמה שקיבלנו לא עבדה. לאחר בחינה מחודשת ודיבוג הבנו שרק 16 בתים מהסיסמה מפוענחים, ולכן עלינו לפרק את הסיסמה לשניים ולפענח את שני החלקים בנפרד.

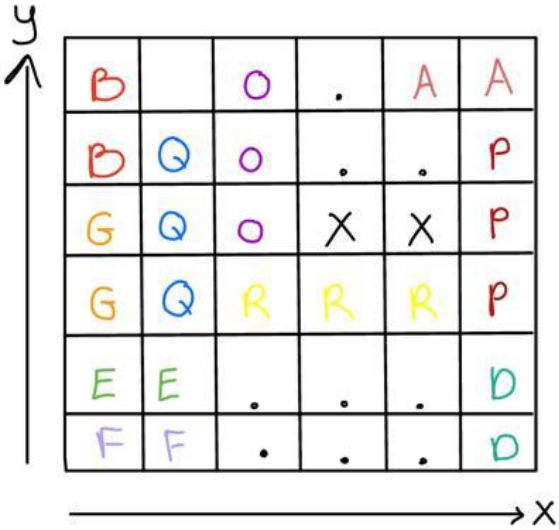
לאחר קבלת הסיסמאות המפוענחות, שרשרנו אותן, וניגשנו לכספת באתר בעזרת הסיסמה.

sheep

בכספת מצאנו קובץ pdf ותמונה.
השתמשנו במפענח התמונות (analyzer.exe) כדי לקבל מידע על התמונה ולאחר מכן הורד קובץ אותו התחלנו לחקור.

מצאנו את פונק' ה-main, ובה קריאה לשתי פונק'.

בפונק' הראשונה התבצעו קריאות ל-memset והשמה של ערכים בזכרון – מה שעורר בנו זכרונות קשים מהמבוך של ה-goblin. ראינו כי בקריאה ל-memset אותחלו 36 בתים, ולכן הסקנו כי גודל הלוח הוא 6X6.
מיד ביצענו דיבוג על-מנת לקבל תמונה של מרחב הזכרון, וראינו את התמונה הבאה:



וראינו שגודל הלוח ורצפי האותיות שבו אכן מסתדרים וכי הניתוח כנראה נכון עד כה.

עברנו לפונק' השנייה, בה המשחק מתנהל בפועל.
מצאנו כי קיימת לולאה שמאפשרת 9 צעדי משחק (בפונק' האתחול אותו משתנה שמשמש כ-counter לצעדים מאותחל ל-9).
כמו כן ראינו את תנאי העצירה ללולאה, שמגדיר מהי נק' ההתחלה ונק' היעד של השחקן שלנו.

התחלנו לקרוא את הגדרת הפונק' וראינו שיש קליטה של 3 תווים בכל איטרציה, ובעזרתם מתבצעים הדברים הבאים:

אחד המשתנים הנסרקים מגדיר אות שמומרת למספר בעזרת פונק' ייעודית, ובעזרת מספר זה אנו ניגשים למערך בזכרון ומחלצים ממנו מידע (4 משתנים של מידע).

לאחר ניתוח מעמיק הצלחנו לפענח מהם המשתנים ומהו המשחק – המשחק הוא זה שבו צריך להוציא את המכונית האדומה (X במקרה שלנו) מלוח המשחק).
4 המשתנים שחולצו בעזרת המספר (שבעצם מגדיר מכונית על הלוח) הם: מיקום אופקי של המכונית, מיקום אנכי של המכונית, אורך המכונית, והכיוון בו המכונית מונחת.

כמו כן, בעת הסריקה נסרק הכיוון בו נרצה להזיז את המכונית הנבחרת (שצריך להתאים לכיוון בו היא מונחת), וכן כמות הצעדים בה נרצה להזיז את המכונית.

בשלב זה, המכונית מוזזת, תוך כדי בדיקה שאכן ההזזה חוקית (אחרת, התוכנית יוצאת).
כמו כן, ראינו כי בכל בחירת צעדים באיטרציה, קיימת פונק' שדואגת לשנות תווים ב-buffer

שמוגדר בהתאם לבחירת הצעדים, והסקנו מנסיון מקדים שכנראה מ-buffer זה תודפס הודעת הסיום, ואכן בסיום הפונק' יש קריאה ל-puts שמבצעת זאת.

לאחר שפתרנו את המשחק, ווידאנו את נכונות הניתוח שלנו קיבלנו פלט לא ברור.
הבנו שישנה חשיבות לסדר הפעולות הנבחרות – והתחלנו לנסות דרך נוספת. בשלב זה ראינו כי הפלט מתחיל להיות הגיוני (מופיעות מילים שלמות, ולא ג'יבריש, אך עדיין לא משפט שלם). בגלל שזיהינו שיש דרך יחידה לפתרון החידה, וההבדל היחידי האפשרי הוא סדר הפעולות בפתרון זה, החלטנו לנסות ידנית לשנות את סדר הפעולות בפתרון.
לבסוף כשקיבלנו את הפלט הנכון, הלכנו לאתר וניסינו להזין את הקוד הסופי שקיבלנו בלוח המשחק של אחד השחקנים (שם הופיעה תיבה המחכה להזנת קוד), אך עדיין זה לא עבד.

נזכרנו שבסריקת קוד ה-QR שביצענו קודם לכן, הופיעה סיומת של "A3-" שלא השתמשנו בה עד כה, וכיוון שייחלנו שהתרגיל מסתיים ברגע זה, קיווינו כי זה המקום היחידי והאחרון שקוד זה ישמש אותנו בו, ולכן שרשרנו את סיומת זו אל הסיסמה שקיבלנו, ואכן הסיסמה התקבלה באתר.

בשלב זה התרגשנו לראות את הכבשה מרקדת, עד כדי כך שניסינו להפעילה שוב, אך לצערנו תם הטקס.