מבנה מחשב 89-230
תשפ"ג 2023

תרגיל 5

הנחיות הגשה

1. העבודה היא ביחידים.
2. ההגשה היא עד ליום ראשון, בתאריך 22.1.2023 בשעה 23:30.
3. קראו בעיון את ההוראות.
4. בהצלחה

רקע כללי

בתרגיל זה הינכם נקראים לדגל – Dr.Evil הטמין "פצצות בינאריות" (binary bombs) על המחשבים שלנו. נבחרתם להיות חלק מכוח המשימה שינטרל את הפצצות, הודות למומחיות שלכם באסמבלי x86-64, ייצוג מידע בזיכרון ופעולות אריתמטיות במחשב. כל אחד מכם יקבל "פצצה בינארית" אותה צריך לנטרל (כל סטודנט את ה"פצצה" שלו). אם לא תעשו זאת, יקרה אחד מהשניים:

1. ציון לא טוב בתרגיל
2. העולם יגיע לקיצו ו-Dr.Evil ינצח

אנחנו לא בטוחים מה מביניהם יקרה, אך ממליצים שתבחרו באחד מן השניים בתור המוטיבציה שלכם לתרגיל :)

מהי פצצה בינארית?

פצצה בינארית היא תוכנית שמורכבת ממספר שלבים (phases). בכל שלב צריך להכניס מחרוזת מסוימת דרך ה-stdin. אם הוכנסה מחרוזת הנכונה עבור שלב מסוים, השלב מנוטרל (defused), והפצצה עוברת לשלב הבא. אחרת, הפצצה מתפוצצת (explodes) – מדפיסה "BOOM!" והתוכנית מסתיימת.
כל שלב בודק היבט אחר של תוכניות בשפת מכונה:

1. שלב 1- השוואות
2. שלב 2- לולאות
3. שלב 3- conditionals/switches
4. שלב 4- קריאות רקורסיביות ושימוש במחסנית (stack)
5. מצביעים
6. רשימות מקושרות/מצביעים/structs

ישנו גם "שלב סודי"...

הפצצה מנוטרלת לאחר שכל אחד מהשלבים נוטרל. כל סטודנט מקבל פצצה ייחודית, כך שיש פתרון (כלומר, רצף המחרוזות שצריך להכניס לתוכנית) שונה לכל פצצה. השלבים מסודרים לפי רמת קושי עולה – אמנם שלב 1 די קל, אבל זה לא אומר שאין עבודה בהמשך.

מבנה מחשב 89-230
תשפ"ג 2023

מכיוון ש-Dr.Evil הטמין כל כך הרבה פצצות, אנחנו נותנים לכל אחד מכם פצצה משלו. המשימה שלכם היא לנטרל את הפצצה לפני התאריך הנקוב בתחילת התרגיל.

בהצלחה וברוכים הבאים ל-Bomb Squad!

איך משיגים את הפצצה שלכם

כל פצצה היא Linux binary executable file שנוצר ע"י קימפול תוכנית בשפת C. כדי להשיג את הפצצה:

1. התחברו למשתמש שלכם בplant
2. הריצו את הפקודה הבאה: get_bomb.sh
3. הכניסו את הפרטים שלכם כפי שהסקריפט מבקש וחזרו לחלון המקורי
4. קובץ הפצצה יירד אל התיקייה הנוכחית שלכם, בשם bombk.tar , כאשר k הוא מספר הפצצה שלכם - מספר ייחודי

תחילת עבודה על הפצצה

התחברו באמצעות ssh ל-plant והגיעו לתיקייה בה שמור הקובץ bombk.tar.
פתחו את ה-.tar באמצעות הפקודה:

```
tar -xvf bombk.tar
```

הפקודה תיצור תיקייה בשם bombk.tar עם הקבצים הבאים:

1. README- זיהוי הפצצה והסטודנט האחראי עליה
2. bomb- קובץ הריצה (executable) של הפצצה
3. bomb.c- קובץ מקור (source file) עם ה-main routine של הפצצה

יתכן וכדי להריץ את הפצצה עליכם לשנות את הרשאות הקובץ bomb.
עשו זאת באמצעות הפקודה:

```
chmod 755 bomb
```

*נטרול הפצצה*בחלון החדש שנפתח

כאמור, המשימה היא לנטרל את הפצצה ע"י הכנסת הקלטים הנכונים לכל 6 השלבים.
כדי לעשות זאת, יש להתחבר למשתמש שלכם ב-plant (הוראות כיצד ניתן להתחבר לשרת מרחוק - https://support.esc.biu.ac.il/) ולהריץ את הפצצה – אין אפשרות להריץ את הפצצה על המחשב האישי אלא רק על גבי ה-plant. בכל פעם שהפצצה מתפוצצת, התוכנית שולחת עדכון למערכת ו**מאבדים נקודה מציון התרגיל**. לאורך כל התרגיל אתם יכולים לעקוב אחרי ההתקדמות שלכם באמצעות scoreboard של התרגיל שנמצא כאן: https://u.cs.biu.ac.il/~89-230

שווי השלבים 1-6 הם 72 נקודות. לכן, אם רק ניסיתם לנטרל את הפצצה, כבר קיבלתם 28 נקודות במתנה!

רמת הקושי של השלבים עולה עם ההתקדמות בתרגיל, אבל כמובן שהמיומנות שלכם תגדל ככל שתתקדמו בשלבים. בכל זאת, השלבים האחרונים הופכים למאתגרים יותר ולכן הקפידו להתחיל את התרגיל בזמן.

הערות וטיפים

1. בכל הרצה של הפצצה, הפצצה תתחיל מהשלב הראשון.
לכן, אם ניטרלתם את שלבים 1-3, למשל, כדי לנסות לנטרל את שלב 4 עליכם להכניס שוב את הקלטים הנכונים עבור שלבים 1-3
2. הפצצה מתעלמת משורות ריקות (blank lines).
אם מריצים את הפצצה עם command line argument, למשל

```
linux> bomb psol.txt
```

הפצצה תקרא את שורות הקלט מ-psol.txt עד שתגיע ל-EOF, ואז את הקלטים הבאים תקרא מה-stdin. זה הודות לרחמים שגילה Dr.Evil, שרצה לאפשר למי שינסה לנטרל את הפצצות שלו להימנע מהקלדה חוזרת של פתרונות לשלבים שכבר ניטרלתם (ראו הערה 1)
3. הפקודה objdump –d על הפצצה שלכם תהיה מאוד שימושית כאן – אמנם ה-source code של הפצצה לא בידכם, אבל תוכלו לקבל את ה-disassembly.
כדי לשמור את הפלט של objdump –d לתוך קובץ (שתוכלו למשל לכתוב עליו הערות), תוכלו להריץ את הפקודה הבאה:

```
objdump -d bomb > mydisassembly.txt
```

4. הפקודה objdump –t על הפצצה שלכם תדפיס את ה-symbol table, שם תוכלו לראות שמות של כל הפונקציות והמשתנים הגלובליים בתוכנית ושמות כל הפונקציות להן הפצצה קוראת והכתובות שלהם.
5. הפקודה strings על הפצצה שלכם תדפיס את ה-printable strings בפצצה.
6. את הפקודות objdump –d, objdump –t, strings אפשר להריץ גם על המחשב האישי שלכם
7. כדי להימנע מלפוצץ את הפצצה בטעות, השתמשו בדיבאגר כדי לקבוע breakpoints בתוכנית ולהתקדם צעד-צעד
8. בנוסף, דיבאגר יהיה כלי מאוד שימושי כאן – בדיקת ערכים ברגיסטרים ובזיכרון יתנו לכם מידע חיוני להבנת התוכנית (קראו עוד ב"שימוש ב-gdb")
9. יתכן ותצטרכו להיעזר בטבלת ASCII כדי להמיר ערכים ב-hex לתווים (טבלה בתרגול 1)
10. אם תצטרכו תיעוד של פקודות שונות, תוכלו להיעזר ב-**man-ו apropos** של פקודות. גם **man ascii** יוכל לעזור

מבנה מחשב 230-89
תשפ"ג 2023

הגשת התרגיל

לא צריך להגיש שום קובץ. ברגע שתנטרלו בהצלחה את הפצצה שלכם, התוכנית תעדכן את המערכת.

הערה חשובה!

אם הייתה לכם שגיאה בעת הורדת הפצצה, תוכלו להוריד חדשה. ברגע שהתחלתם לעבוד על פצצה מסוימת – ניקוד ינתן עבור הפצצה. למקרה של שגיאה בעת הורדת הפצצה - ניתן להוריד פצצה חדשה (כלומר, סטודנט יכול להוריד מספר קבצים). כל פצצה היא פצצה רנדומית המוקצת מהשרת לכן, הציון עבור נטרול קבצי הפצצות הניקוד הוא מצטבר.

כלומר אם הורדתם 2 קבצים קיבלתם 2 פצצות שונות לנטרל.
אם בפצצה הראשונה קיבלתם קנס של 1 נק' ובפצצה השנייה קיבלתם קנס של 2 נק', החישוב הסופי בציון יהיה כמו שקיבלתם קנס של סה"כ 3 נק'.

שימוש ב-gdb

1. פקודות שימושיות בדיאבגר מופיעות בקובץ How_To_GDB.pdf
2. display $rdx תציג את תוכן הרגיסטר rdx אחרי ביצוע כל פקודה בתוכנית
3. stepi או si תקדם את התוכנית צעד אחד
4. כדי ש-gdb ידפיס את הפקודה הבאה בכל פעם: display /i $rip (מאוד שימושי!).
   הרגיסטר rip שומר את כתובת הפקודה הבאה בתוכנית שתתבצע.
5. x/4x 0xbfbff0d4c תדפיס 4 מילים (בפורמט hex) שמתחילות בכתובת 0xbfbff0d4c
6. x/s $rdx תדפיס את המחרוזת (רצף תווי ASCII שמסתיימים ב-null) שמתחילה בכתובת השמורה ב-rdx
7. כדי לבצע disassembly לפונקציה שלמה, למשל main:
   disassemble main
8. כדי לקבל מידע על ה-frame הנוכחי במחסנית: info frame
9. כדי להדפיס מידע לגבי כל ה-active stack frames, ניתן להשתמש בפקודה: backtrace

בנוסף ישנה הקלטה של שימוש בGDB בחומרי העזר שהועלתה למודל

בהצלחה!