# ארגון ותכנות המחשב

תרגיל בית 4 (יבש)

המתרגל האחראי על התרגיל: לירן רדל.

הנחיות:

* **שאלות על התרגיל ב- Piazza בלבד.**
* ההגשה **בזוגות**.
* על כל יום איחור או חלק ממנו, שאינו באישור מראש, יורדו 5 נקודות.
  + ניתן לאחר ב-3 ימים לכל היותר.
  + הגשות באיחור יתבצעו דרך אתר הקורס.
* לכל שאלה יש לרשום את התשובה במקום המיועד לכך.
* את התרגיל יש להגיש באתר הקורס **כקובץ pdf.**
* יש לענות **בטופס התרגיל. ניתן לעשות זאת באחת מהאפשרויות הבאות:**
  + להקליד את התשובות במסמך ה-WORD ולבסוף לשמור כ-pdf.
  + לכתוב אותן על גבי גרסת ה-pdf, בעזרת הטאבלט החביב עליכם בכתב ברור וקריא.
  + להדפיס את גרסת ה-pdf ולכתוב על הטופס המודפס את התשובות בכתב יד ברור וקריא.
* תיקונים לתרגיל, אם יהיו, יופיעו ממורקרים.
* יש לפתור את המטלה לפי סדר הפתרון. החלק השני והשלישי יהיו נגישים לאחר פתירת החלק הראשון בלבד.

## חלק ראשון – פיענוח קוד CrackMe (30 נקודות)

**יש להתחיל מחלק זה, מאחר והחלק השני והשלישי מוגנים בסיסמא אותה תפענחו בחלק זה.**

לאחר הבסת זאקום בתרגיל הבית הקודם, ניק ועדן נתקלו בדלת סתרים הנמצאת בתוך המאגמה של זאקום. על הדלת כתוב "ענו על שלושת החידות והכנסו". בתוך המאגמה שמו לב לקובץ .out שבעצם מקבל 3 קלטים, כל קלט הינה חידה, והם הבינו כי עליהם לכתוב 3 קלטים כדי לקבל את הסיסמא שתפתח את דלת הסתרים שמצאו.  
  
ברשותכם קובץ KeysToRoom.out, ותוכלו לשים לב שאם תנסו לפתוח אותו בעזרת הכלים הרגילים שברשותכם, תקבלו פלט שנראה מאוד מוזר.

בעזרת הפקודה objdump -d הנלמדה בקורס שעוזרת לפענח את קוד הקובץ, תוכלו לראות שהוא בנויי מהרבה חלקים בתוכו. ניק ועדן החליטו לעבוד שלב שלב בנסיון לפרוץ את הקובץ ולחקור אותו.

### סעיף 1 (4 נקודות)

באיזה כתובת מתחילה \_start ו-main?

כתובת main הינה (בבסיס הקסהדצימלי): **\_\_\_\_\_\_400537\_\_\_\_\_\_\_**0x.

כתובת \_start הינה (בבסיס הקסהדצימלי): **\_\_\_\_\_\_\_400450\_\_\_\_\_\_\_\_**0x.  
  
עדן וניק שמו לב כי הקוד שנמצא החל מ-main, מכיל 3 קריאות של scanf, שמסתדר עם העובדה כי הקוד מצפה לקבל 3 קלטים, וכל פעם על הקלט שנקלט מבוצעת בדיקה, ואם הבדיקה עוברת ממשיכים לשלב הבא.

**הבהרה לגביי הסעיפים הבאים:**

הפונקציה scanf מקבלת 2 פרמטרים: הפרמטר הראשון הוא ה-format string, עדן גילה שהוא בעצם %d לאחר חקירה עצמית שעובר בכל אחד מה-scanf, והפרמטר השני הוא המיקום של המשתנה אליו ייכתב הערך שנקלט.

**המשך התרגיל בעמוד הבא**

### סעיף 2 (7 נקודות)

מהי הבדיקה הראשונה המתבצעת על קלט ה-scanf הראשון? ואיזה קלט יפתור אותה? נמקו

**ההשוואה מתבצעת באמצעות הפקודה "test %eax, %eax" בכתובת 40055a ולאחר מכן כתובה פקודת jle הקופצת למקטע שמוציא אותנו מהתוכנית. כאשר %eax מכיל את ערך המספר הראשון הוא הקלט של scanf. פקודת test תבצע AND על %eax עם עצמו ולכן תקבע את הדגלים OF=CF=0 ותעדכן את SF, ZF,PF בהתאם לערך %eax. בהתאם, פקודת jle קופצת אם ורק אם מתקיים: ZF=1 או SF != OF.**

**מכיוון ש-OF=0 הקפיצה תתרחש אך ורק אם %eax <= 0. לכן הבדיקה הראשונה היא האם %eax > 0. כלומר האם הקלט הראשון חיובי.**

**בעצם בודקים אם מתחלק ב-7... וטפ...**

### סעיף 3 (6 נקודות)

מהי הבדיקה השנייה המתבצעת על קלט ה-scanf השני? ואיזה קלט יפתור אותה? נמקו

**בודקים האם הקלט שווה 47.**

### סעיף 4 (8 נקודות)

מהי הבדיקה השלישית המתבצעת על קלט ה-scanf השלישי? ואיזה קלט יפתור אותה? נמקו

**בודקים אם 20.**

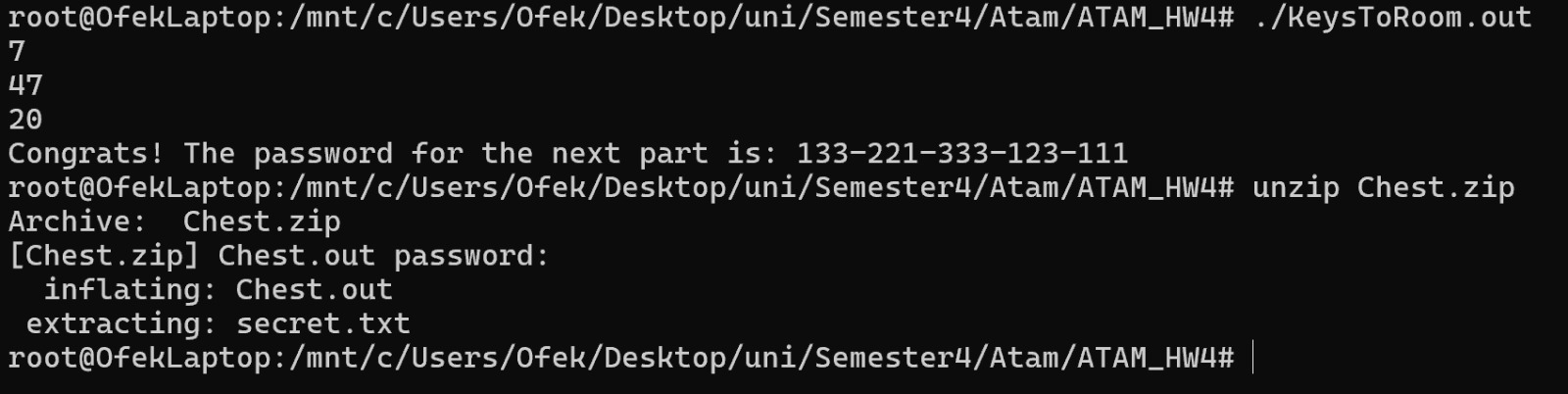
**המשך התרגיל בעמוד הבא**

### סעיף 5 (5 נקודות)

**תנו דוגמא לקלט** שיגרום לתוכנית להדפיס את הסיסמא לחדר הסתרים. מהי הסיסמא?

**יש לצרף צילום מסך** של ההדפסה.

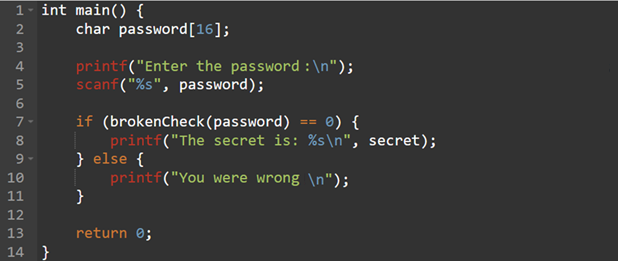
**הסיסמא**: 133-221-333-123-111  
 **צילום מסך של ההדפסה:**



**המשך התרגיל בעמוד הבא**

## חלק שני – חולשת BufferOverFlow (20 נקודות)

**השתמשו בסיסמא שמצאתם כדי לפתוח את קובץ ה-ZIP כדי לפתור את שאר התרגיל.**

לאחר שניק ועדן הצליחו בעזרת הסיסמא שמצאתם עבורה לפתוח את דלת הסתרים, הם מצאו תיבה שכמובן ניסו לפתוח אותה, ללא הצלחה. הם גילו שגם התיבה נעולה בסיסמא, שעל מנת שיוכלו לפתוח אותה, הם צריכים להזין את הסיסמא כדי שתיפתח. בעזרת לבישת קדסת הזאקום, שנתנה להם הרבה חוכמה, עדן הצליח להשיג חלק מן קוד המקור של המנעול של התיבה: (secret אינו מוגדר ב-main ותוכנו אינו ידוע)  
  
****  
  
אמנם לניק ועדן אין את הסיסמא לפריצת המנעול, אך עדן שם לב לחולשה בקוד המקור אותה ניתן לנצל.

ניתן לקרוא על החולשה כאן: <https://en.wikipedia.org/wiki/Buffer_overflow>

### סעיף 1 (5 נקודות)

ניתן לראות כי לפי הקוד, המשתמש מצפה לקבל סיסמא באורך 16 תווים, ובעצם בעזרת הפונקציה scanf קורא את הסיסמא שהמשתמש מכניס אל תוך הבאפר password. מה הבעיה בשימוש ב-scanf בקוד זה?

**הבעיה ב-scanf היא ש-password מוקצה על המחסנית בגודל 16 בתים, ומכיוון שהשתמשנו ב-scanf ולא ב-fgets אנו עלולים לקבל יותר מ-16 בתים ולכתוב אותם ל-password דבר שיוביל לגלישה של זיכרון במחסנית, ובפרט לדריסה של כתובת החזרה מ-scanf השמורה במחסנית.**

**המשך התרגיל בעמוד הבא**

### סעיף 2 (5 נקודות)

ניק ועדן התנסו עם התוכנה, וניסו להכניס את הסיסמה: “maplestoryisthebestnostalgicgame!!”.

לאיזו כתובת תקפוץ פקודת ret שמבצעת הפונקציה main בסיום ריצתה?

רמז: מה אורך הקלט שהכניסו ניק ועדן? על מה הדבר משפיע?

**הבא ננתח את המחרוזת שהוכנסה החל מהכתובת של password במחסנית (אות ראשונה) עד הכתובות הגבוהות יותר בהן שמורה כתובת החזרה.**

**הולך לתוך הססמא: maplestoryistheb (16 בתים חוקיים)**

**הולך לתוך rbp: estnosta (8 בתים שדורסים את rbp השמור מתחת לכתובת החזרה ומעל למערך)**

**הולך לתוך כתובת החזרה: lgicgame (8 בתים שדורסים את כתובת החזרה השמורה מעל rbp)**

**זבל: !! (2 בתים של ערכי זבל שדורסים את ה-frame הקודם שהוקצה במחסנית)**

**נתרגם את כתובת החזרה בהתאם.**

**נשים לב כי האות הראשונה שמורה בכתובת נמוכה והאות האחרונה שמורה בכתובת גבוהה. מכיוון שהכתובת מיוצגת ב-little endian נקבל כי כתובת החזרה היא:**

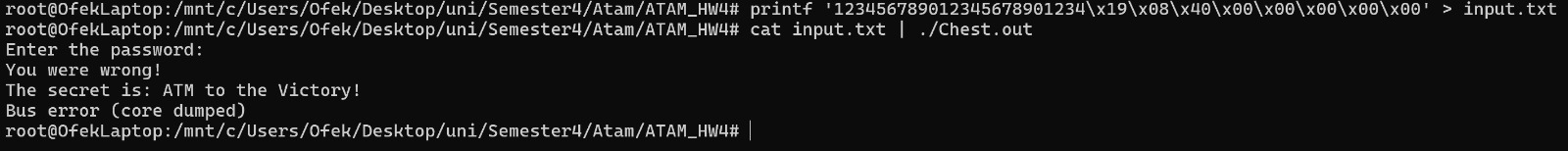
**0x65 6D 61 67 63 69 67 6C**

### סעיף 3 (10 נקודות)

**תנו דוגמא לקלט** שיגרום לתוכנית Chest.outלהדפיס את secret, כלומר לבצע את שורה 8 בקוד ה-C. **יש להסביר בקצרה** את דרך פעולתכם כולל ניתוח הקלט שבחרתם. בתשובתכם השתמשו בפורמט \xHH כדי לציין קלט הקסהדצימלי. לדוגמא אם הקלט הינו האות a ואחריה בית עם ערך 0x8F שאחריו 0x90, כתבו “a\x8F\x90”.

**יש לצרף צילום מסך** של ההדפסה. ייתכן וכי יהיו הדפסות נוספות שיקרו, וכי התוכנית לא תסתיים כראוי לאחר ההדפסה.   
מומלץ להשתמש בכלי echo כדי להזין את הקלט לתוך הקובץ Chest.out.

**קלט לתוכנית**: **123456789012345678901234\x19\x08\x40\x00\x00\x00\x00\x00  
  
הסבר**: **24 הבתים הראשונים (המספרים) ימלאו את 16 הבתים שהוקצו ל-password ו-8 הבתים במחסנית עליהם שמור rbp. שמונה הבתים האחרונים הכתובים בבסיס הקסאדצימלי הם כתובת החזרה (כתובת prinf שמדפיס את secret) ב-little endian: האות הראשונה היא התחילת הכתובת (כתובת החזרה היא: 0x00000400819).**  
 **צילום מסך של ההדפסה:**



**המשך התרגיל בעמוד הבא**

## חלק שלישי – התקפת ROP וחקירת קובץ ELF (50 נקודות)

לאחר פתיחת התיבה, ניק שם לב כי יש קובץ ישן בתוך המגמה של זאקום, שאינו מצליח לפתוח את תוכנו. עדן בחוכמתו הבין כי אנו יכולים להשתמש בחולשה שמצאנו כדי לנסות לקרוא את התוכן של הקובץ הזה, באופן זדוני ועקיף. (אמנם הקובץ נתון לכם, אך ניתן להניח לשם הסיפור והמתקפה כי אינו נגיש אליכם ישירות).  
  
ראשית, כדי לנסות לנצל את חולשת הקוד אפילו יותר ואיך לנצלה, נחקור את הקובץ Chest.out אפילו יותר לעומק, בעזרת הידע שלמדנו בקורס (כגון readelf, objdump וכדומה).

### סעיף 1 (6 נקודות)

1. מה גודל ה-Section Header Table של התוכנית? **\_\_\_\_\_\_\_\_\_\_\_\_\_**
2. כמה Program Headers מוגדרים בקובץ? **\_\_\_\_\_\_\_\_\_\_\_\_\_**
3. מה גודל כל כניסה ב-Program Header Table? **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**
4. עבור כל program header מסוג LOAD, הכניסו את נתוניו לטבלה הבאה (יתכנו שורות ריקות):

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **הרשאות (סמנו את ההרשאות)** | **כמה bytes ייקראו מהקובץ ל-segment** | **גודל שיתפוס בזיכרון הווירטואלי** | **כתובת טעינה לזיכרון הווירטואלי** | **מיקום בקובץ (offset בבתים)** |
| R W E | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** |
| R W E | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** |
| R W E | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | 0x**\_\_\_\_\_\_\_\_\_\_\_\_** |

1. באיזו כתובת יתחיל הקוד את ריצתו? 0x**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**המשך התרגיל בעמוד הבא**

### סעיף 2 (7 נקודות)

עדן וניק הבינו כי על מנת לבצע את המתקפה שהם מתכננים, הם ניסו למצוא את קטעי הקוד הבאים בתוך Chest.out.

1. עבור כל שורה מלאו את הכתובת בזיכרון התוכנית שבו נמצא קידוד הפקודות. אם הקידוד מופיע בכמה אזורי זיכרון, בחרו באזור בעל הרשאות הרצה.  
   רמז: מומלץ לקרוא על הכלי grep ולהשתמש בכלים כגון objdump ו-hexdump, שנלמדו בתרגול 8.

|  |  |
| --- | --- |
| **כתובת** | **פקודות** |
| 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | **popq %rdi ret** |
| 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | **popq %rsi ret** |
| 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | **popq %rax ret** |
| 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | **popq %rdx ret** |
| 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | **movq %rax, %rdi ret** |
| 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | **movq %rax, %rdx ret** |
| 0x**\_\_\_\_\_\_\_\_\_\_\_\_** | **syscall  ret** |

**המשך התרגיל בעמוד הבא**

### סעיף 3 (10 נקודות)

עדן וניק רצו לחדד את כישורי האסמבלי שלהם לפני שהם ימשיכו במשימה.

עבור כל שורה מלאו את העמודה הריקה בהתאם לעמודה השנייה. כלומר, לתרגם משפת מכונה לשפת אסמבלי, או לתרגם מספת אסמבלי לשפת מכונה כפי שנלמד בתרגול והרצאה 7.

|  |  |
| --- | --- |
| **הפקודה בשפת מכונה** | **הפקודה בשפת אסמבלי** |
| **ff 25 6b 51 40 00** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **movw 0x2316ae(%rip), %dx** |
| **48 8b 04 25 fe 15 23 61** | **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **popq %rbx**  **leave ret** |
| **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** | **movq %r11, %rbx** |

**המשך התרגיל בעמוד הבא**

### סעיף 4 (2 נקודות)

עדן הצליח בעזרת החוכחה שקיבל מקסדת הזאקום, להבין כי קיימים 2 משתנים שיכולים לעזור לנו בתכנון המתקפה:



כתובת fname הינה (בבסיס הקסהדצימלי): **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**0x.

כתובת readbuf הינה (בבסיס הקסהדצימלי): **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**0x.

### סעיף 5 (5 נקודות)

כעת עדן וניק טוענים שבידם כל המידע הנדרש כדי לבצע את המתקפה, ולהשיג את המידע הסודי הקיים ב-secret.txt.  
קראו על התקפת ROP - <https://en.wikipedia.org/wiki/Return-oriented_programming>.

באילו קריאות מערכת הנלמדו בקורס עדן וניק יכולים להשתמש על מנת לפתוח את הקובץ, לקרוא את תוכנו ולהדפיסו למסך? נמקו

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**המשך התרגיל בעמוד הבא**

### סעיף 6 (20 נקודות)

כעת ברצונינו לבצע את ההתקפת ROP, כדי להשיג את תוכן הקובץ הנסתר! (שתוכנו לא עולה על 100 תווים).

**תנו דוגמא לקלט** שיגרום לתוכנית Chest.outלהדפיס את תוכן הקובץ secret.txt לערוץ הפלט הסטנדרטי. **יש להסביר בקצרה** את דרך פעולתכם כולל ניתוח הקלט שבחרתם. בתשובתכם השתמשו בפורמט \xHH כדי לציין קלט הקסהדצימלי. לדוגמא אם הקלט הינו האות a ואחריה בית עם ערך 0x8F שאחריו 0x90, כתבו “a\x8F\x90”.

**יש לצרף צילום מסך** של ההדפסה. השתמשו בפתרון שלכם בכל המידע שמצאנו – חלק מה-gadgets שמצאנו בסעיף 2 יכולים לעזור לכם בבניית שרשרת ההתקפה, הכתובות שמצאתם בסעיף 4 יכולות לעזור לכם בקריאות המערכת אותם אתם עלולים לבצע, וסעיף 5 יכול לתת לכם כיוון לאיך לבצע את ההתקפה.  
הבהרה – התוכנית עלולה להסתיים לא בצורה תקינה, לא ירדו על כך נקודות, כל עוד תוכן הקובץ מודפס למסך (אתם יוצאים מנקודת הנחה שאינכם יודעים את תוכנו). קוד שמסתיים בצורה תקינה ייקבל בונוס של נקודה.  
מומלץ לשמור באותה תיקייה את secret.txt ואת Chest.out כדי לבצע את ההתקפה.

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

**צילום מסך של ההדפסה:**

בהצלחה!