מס' תעודת זהות: _ _ _ _ _ _ _ _ _ _ מס' מחברת: _ _ _

אוניברסיטת תל-אביב
מדעי המחשב

מבוא לאבטחת מידע
תשע"ד סמסטר ב' מועד א'
25.6.2014
(כולל תיקונים והבהרות שניתנו בזמן המבחן)

פתרון המבחן

מרצים: ערן טרומר ואבישי וול

הנחיות:

- **משך הבחינה 3 שעות.**
- **מותר שימוש בכל חומר עזר כתוב/מודפס על גבי נייר.**

- משקל כל שאלה מצוין בתחילתה. יש לענות על כל השאלות.
- תשובה ללא נימוק לא תזכה בניקוד.
- כתבו את תשובותיכם על גבי טופס המבחן במקום המוקצה לכך. המקום הוקצה כך שיספיק לתשובה מלאה בכתב יד מרווח. תשובות במחברת הבחינה לא יבדקו. מומלץ לכתוב תחילה טיוטת תשובה במחברת הבחינה, ורק אז להעתיק אותה, בצורה ברורה וקריאה, לטופס המבחן.
- ניתן לענות בעברית או באנגלית.

- במבחן זה **13** עמודים (כולל עמוד זה). אנא ודאו שכולם ברשותכם.

ב ה צ ל ח ה!

לשימוש הבודקים:

1. **(א)**	2. **(א)**	3. **(א)**	4. **(א)**	5. **(א)**	6. **(א)**
		(ב)			
(ב)	**(ב)**	**(ג)**	**(ב)**	**(ב)**	**(ב)**
		(ד)			
(ג)	**(ג)**	**(ה)**			**(ג)**
		(ו)			
	(ד)	**(ז)**			
		(ח)			
	(ה)	**(ט)**			

מס' מחברת: _ _ _

שאלה 1 (10 נק')

ישנן מערכות הרשאה העובדות בשיטת access-control-lists ומערכות שמשתמשות ב-capabilities. ציינו יתרון אחד וחיסרון אחד למערכות מבוססות capabilities.

א. [3 נק'] יתרון:

סובייקט המחזיק ב-capability יכול בקלות להעניק אותו הלאה להתהליך אחר שהוא בוטח

בו, כדי להאציל סמכויות.

כמו כן, תהליכים שונים של אותו משתמש יכולים להחזיק בקבוצה שונה של capabilities,

---

בהתאם לעקרון ה-least privilege.

ב. [3 נק'] חסרון:

לאחר שהוענק capability, לא ניתן לבטל אותו (ללא מנגנונים נוספים).

כמו כן, בגלל שכל תהליך יכול להעניק הלאה את ה-capabilities שלו, קשה לנהל מרכזית

למי יש גישה למשאב (אובייקט) נתון.

ג. [4 נק'] האם מערכת ההרשאות של Android היא מסוג access-control-lists או capabilities ? נמקו.

Android מנהל רשימת הרשאות לכל אפליקציה (כלומר שורות)

ב-access control list, כמו ב-capabilities. אבל חסרה בו יכולת מהותית של מערכות

---

capabilities: האצלת סמכויות בין תהליכים.

מס' מחברת: _ _ _

שאלה 2 (20 נק')

בארגון מסויים, כל פעם שמשתמש מתחבר לרשת הארגונית האלחוטית Public-PSI, בפעם הראשונה שהוא פותח בדפדפן כתובת `http://...` כלשהי, מופיע דף המבקש ממנו להקליד את שם המשתמש הארגוני וסיסמתו כדי להיכנס לרשת. שורת הכתובת של הדפדפן מציגה את האתר המבוקש (לדוגמה `http://google.com`), אבל גוף הדף מכיל את טופס ההזדהות הארגוני. רק אחרי הזנת שם וסיסמה תקינים, הרשת עובד כרגיל.

א. [4 נק'] כיצד לדעתך ממומשת המערכת?

הארגון עושה מניפולציה לתרגום DNS : לא משנה איזה שם אתר המשתמש כתב ב-browser,

אם אין רישום שהמשתמש כבר הזדהה - שרת ה-DNS מחזיר לו כתובת IP של שרת

ההזדהות. המערכת צריכה לזכור את כתובת ה-ip (או כתובת MAC) של המשתמש אחרי

שהוא מזדהה .

ב. [4 נק'] תארו סכנת בטיחות בשיטה זו.

מספר תשובות אפשריות. למשל: (1) הכל ב-http כולל טופס ההזדהות (לא צויין בשאלה אבל

נניח שכך) אז הסיסמא חשופה .(2) ARP poisoning מאפשר למשתמש פנימי אחר להזריק את

עצמו לאמצע הקשר. (3) תלוי איך המערכת זוכרת מי כבר הזדהה, אבל אם זה מבוסס כתובת IP

אז אפשר ע"י האזנה לרשת למצוא כתובת IPשל משתמש אחר, להשתלט עליה, ולגלוש.

מס' מחברת: _ _ _

ג. [4 נק'] אם, לאחר ההתחברות לרשת (אבל לפני הזנת השם והסמה), הדף הראשון שהמשתמש פותח בדפדפן הוא בכתובת `https://...` כלשהי, אזי מופיעה בדפדפן התראה סטנדרטית מסויימת, עם אפשרות [המשך] ו-[עצור]. איזו התראה מופיעה ומדוע?

יהיה חוסר התאמה בין הסרטיפיקט שיציג שרת ההזדהות לבין שם האתר המבוקש – שרת

ההזדהות לא יכול להציג סרטיפיקט לגיטימי של gmail.com – ושם האתר נמצא וחתום בתוך

הסרטיפיקט. כתוצאה מכך ה-browser יתלונן.

ד. [4 נק'] האם מנהלי הרשת הארגונית יכולים למנוע את הופעת ההתראה?

לא ב-browser סטנדרטי . מבחינת הדפדפן, זה נראה בדיוק כמו התקפת

---

man-in the-middle שמנגנון SSL/TLS נועד להתריע עליו.

ה. [4 נק'] מנהלי הרשת הוציאו הנחייה שבמקרה הנ"ל, צריך פשוט לאשר [המשך] ואז מופיע הדף הרגיל להזנת השם והסיסמה. מה הבעיה בהנחיה זו?

מאמן את המשתמשים להתעלם מהתראה משמעותית של הדפדפן.

מס' מחברת: _ _ _

שאלה 3 (40 נק')

RLE (Run-Length Encoding) היא שיטת כיווץ מידע פשוטה, שבה דוחסים רצפים של בתים זהים ע"י סימון בתו מיוחד `0xff` ואחריו מספר החזרות. התו `0xff` עצמו מיוצג כ-`0xff,0xff`.

להלן כמה דוגמאות לקלט ופלט של תהליך הדחיסה (המחרוזות משמאל לימין, ותווים שאינם דפיסים מיוצגים ע"י `0x##` כאשר ## הינו הערך ההקסהדסימלי שלהם):

```
a,b,c  ➔  a,b,c
a,b,c,q,q,q,q,q,q,x,y,z       ➔ a,b,c,0xff,0x06,q,x,y,z
a,b,c,q,q,q,q,q,0xff,x,y,z ➔ a,b,c,0xff,0x06,q,0xff,0xff,x,y,z
```

בספריה נפוצה, מצאתם את הפונקציה הבאה הממשת פריסה (decompression) של דחיסת RLE:

```
// Function: RLE_unpack
// Unpacks the RLE coded data from 'in' to 'out'.
// 'out_len' specifies the maximum number of characters that
// can be written to 'out', and is updated with the number of
// characters actually written.
void RLE_unpack(unsigned char *out,  int *out_len,
                unsigned char *in,   int in_len)
{
     int i=0;
     int j=0;

     while (i<*out_len && j<in_len) {
          if (in[j]!=0xff || j==(in_len-1))
          {
               out[i++] = in[j++];
          }
          if (in[j]==0xff && j<(in_len-2) && in[j+1]!=0xff)
          {
               int k;
               for (k=0; k<in[j+1] && i<*out_len; k++)
               {
                    out[i++] = in[j+2];
               }
               j+=3;
          }
          if (in[j]==0xff && j<(in_len-1) && in[j+1]==0xff)
          {
               out[i++] = 0xff;
               j += 2;
          }
     }
     *out_len = i;
}
```

מס' מחברת: _ _ _

במהלך בדיקת fuzzing נתגלה כי כאשר קוראים ל-

```
RLE_unpack(tmpbuf, &tmpbuf_len, inbuf, 5)
```

עם `tmpbuf_len=100` ו-`inbuf` מצביע למחרוזת `0xff,0xfe,0x41,0xff,0xff`,
אזי ביציאה `tmpbuf_len=101`.

א. [2 נק'] מה הבית שנכתב מעבר לקצה של `tmpbuf`?

0xff

ב. [5 נק'] הציעו תיקון לבעיה בקוד.

יש מס' דרכים, הכי פשוטה, היא הוספת else לif השלישי, זה גורם לבדיקת האורך בין ה-if-ים.

כך נמנעת הכתיבה הנוספת שנוצרת כאשר יש רצף 0xff,size,char,0xff,0xff

---

ג. [3 נק'] הסבר מהו bit-flip fuzzing.

שיטה המשמשת למציאת חולשות ותקלות בתוכנה ע"י לקיחת קלט תקין והזנתו לתוכנית כל

פעם כאשר בכל איטרציה משנים ביט אחד מהקלט התקין ומריצים את התוכנה ובודקים האם

התוכנית מתנהגת כראוי בכל ריצה שכזו.

מס' מחברת: _ _ _

להלן קוד IDA של פונקציה השייכת לתוכנה המשתמשת ב-RLE_unpack. בהנחה שהמתקיף שולט בקלט לפונקציה, עקוב אחר ההוראות כדי לדרוס הרבה יותר בתים באמצעות החולשה:

```
.text:080484D8                          ; ||||||||||||||| S U B R O U T I N E |||||||||||||||||||||||||||||||||||||||
.text:080484D8                          ; Attributes: bp-based frame
.text:080484D8                          sub_80484D8    proc near               ; CODE XREF: .text:0804835Bp
.text:080484D8
.text:080484D8                          var_70         = dword ptr -70h
.text:080484D8                          var_C          = dword ptr -0Ch
.text:080484D8                          arg_0          = dword ptr  8
.text:080484D8                          arg_4          = dword ptr  0Ch
.text:080484D8
.text:080484D8    55                                   push    ebp
.text:080484D9    89 E5                                mov     ebp, esp
.text:080484DB    83 EC 78                             sub     esp, 78h        ; Integer Subtraction
.text:080484DE    FF 75 0C                             push    [ebp+arg_4]
.text:080484E1    FF 75 08                             push    [ebp+arg_0]
.text:080484E4    8D 45 F4                             lea     eax, [ebp+var_C] ; Load Effective Address
.text:080484E7    C7 45 F4 64 00 00 00                 mov     [ebp+var_C], 64h
.text:080484EE    50                                   push    eax
.text:080484EF    8D 45 90                             lea     eax, [ebp+var_70] ; Load Effective Address
.text:080484F2    50                                   push    eax
.text:080484F3    E8 3C FF FF FF                       call    RLE_unpack      ; Call Procedure
.text:080484F8    58                                   pop     eax
.text:080484F9    5A                                   pop     edx
.text:080484FA    FF 75 F4                             push    [ebp+var_C]
.text:080484FD    68 E0 85 04 08                       push    offset aUnpacked_lenD ; "unpacked_len: %d\n"
.text:08048502    E8 F9 FD FF FF                       call    _printf         ; Call Procedure
.text:08048507    83 C4 10                             add     esp, 10h        ; Add
.text:0804850A    C9                                   leave                   ; High Level Procedure Exit
.text:0804850B    C3                                   retn                    ; Return Near from Procedure
.text:0804850B                          sub_80484D8    endp
```

ד. [8 נק'] תארו את פעולת הפונקציה בצורה קצרה ומדויקת. הסבירו מהם שני הפרמטרים שהיא מקבלת.

הפונקציה מקבלת מחרוזת תווים(arg0) ואורך (arg4). הפונקציה מדפיסה את האורך של הקלט המכווץ כאשר

הוא פרוש (Unpacked). על מנת לעשות זאת, היא מקצה באפר על המחסנית בגודל 100 שאליו ייפתח

הקלט באופן זמני, בנוסף על המחסנית מוחזק האורך של המחרוזת הפרושה. אפשר גם לרשום כך:

---

void print_unpack_len(char *in, int in_len) { int tmpbuf_len = 100; int tmpbuf[100]; RLE_unpack(tmpbuf, &tmpbuf_len, in, in_len);

printf("unpacked len: %d", tmpbuf_len); }

מס' מחברת: _ _ _

ה. [5 נק'] תארו את מבנה המחסנית של הפונקציה עד לכתובת החזרה של הפונקציה sub_80484D8 בדיוק לפני הקריאה לפונקציה RLE_unpack (לאחר ביצוע הפקודה ב-.text:080484F2):

תוכן	אורך בבתים	Offset from ebp
arg_0	4 Bytes	+8
return address	4 Bytes	+4
ebp	4 Bytes	0
int tmp_outlen	4 Bytes	-0xc (skips 8 unused)
char buf[100]	100 Bytes	-0x70
(int) Copy of arg_4	4 Bytes	-0x7c (skips 8 unused)
(char *) Copy of arg_0	4 Bytes	-0x80
(int *) &tmp_outlen	4 Bytes	-0x84
(char *) buf	4 Bytes	-0x88

ו. [3 נק'] הסבר כיצד ישפיע הקלט מראש השאלה על המחסנית ומה האפקט על ריצת הפונקציה RLE_unpack.

מחוץ להקשר זה היה לנו רק off-by-one פשוט, כעת מכיוון שהמחסנית מסודרת כך שtmp_outlen מופיע בסוף

הבאפר וגם אנחנו נמצאים במעבד little-endian, שאומר שהבית הראשון הוא הקטן יותר ואינו מכיל את הסימן,

---

נוכל לשנות את האורך תוך כדי ריצת הפונ' ולמעשה לאפשר דריסה שרירותית שתעצר רק כתלות בקלט.

ז. [8 נק'] הניחו שהמתקיף קובע את את המחרוזת לפריסה ע"י הפונקציה. המשיכו את המחרוזת מסעיף ב' כדי ליצור shellcode מסוג ROP אשר מפעיל shell ואינו גורם לקריסה. הניחו שידועות הכתובות הבאות:

```
0B7F7FF18h - "/bin/sh"
0B7E5E430h – system(char *cmd)
0B7E51FB0h – exit(int errcode)
```

מלאו את הטבלה בערכים ההקסדצימליים של המחרוזת שלכם משמאל לימין:

ff	fe	41	ff	ff	ff	10	30	e4	e5	b7	b0	1f	e5	b7	18	ff	ff	f7	b7

מס' מחברת: _ _ _

ח. [3 נק'] לאחר גילוי הבאג, מנהל המערכת כיבה את מנגנון ה-DEP. האם ה-shellcode עדיין יעבוד, ומדוע?

כן. מנגנון Data Execution Prevention, מונע הרצה של קוד ממקומות אליהם ניתן לכתוב כגון

המחסנית. שיטת ROP בה השתמשנו ממילא באה לעקוף בעיה זו ע"י שימוש בקטעי קוד

הנמצאים בזכרון המוגדר כמותר לריצה. ולכן כיבוי המנגנון היה חסר משמעות לחלוטין

ט. [3 נק'] כעת מנהל המערכת הפעיל את מנגנון ה-ASLR. האם ה-shellcode עדיין יעבוד, ומדוע?

לא. Address Space Layout Randomization גורם להגרלת הכתובות במרחב הזכרון

בעקבות כך הכתובות של הפונ' תהיינה מוגרלות בזכרון ולא תהיה התאמה בינן לבין הכתובות

שהשתמשנו בהן בshellcode.

שאלה 4 (8 נק')

א. [4 נק'] רוב האתרים שמצריכים סיסמת כניסת לא מסוגלים לשחזר סיסמא שנשכחה, ובמקום זאת מאפשרים רק להחליף את הסיסמא (לאחר הזדהות בשיטה אחרת). הסבירו מדוע, טכנית, אין אפשרות שחזור סיסמאות.

האתרים לרוב אינם שומרים סיסמאות, אלא רק תמצית של הסיסמאות תחת פונקציית

תמצות (hash), כדי שגניבת מאגר נתוני חשבונות המשתמשים לא תחשוף את הסיסמאות

שלהם. מאחר והפונקציה היא חד-כיוונית (קשה להיפוך), לתוקף קשה לשחזר את הסיסמה,

---

אבל גם לאתר עצמו קשה.

ב. [4 נק'] תארו בקצרה את התועלת בשימוש ב-salt בשמירת סיסמאות .

salt מקשה חישובית על התקפות מילון לשחזור סיסמאות חלשות. אם האתר משתמש

בתמצית ללא salt, תוקף יכול לחשב מראש, פעם אחת, את התמציות של סיסמאות נפוצות, ואז

לחפש אותן במאגר המשתמשים שגנב (אפשר גם להעזר ב-rainbow table). עם salt, התוקף

יצטרך לעבוד קשה יותר ולחשב את התמציות מחדש עבור כל ערך salt של משתמש וכל

סיסמה נפוצה.

(כמו כן, salt מונע מהתוקף לראות בקלות שלשני משתמשים יש סיסמה זהה.)

(הערה: מדובר בתרחיש בו התוקף גנב את מאגר המשתמשים הכולל, תמציות וגם את ה-salt.

כעת ה-salt ידוע לתוקף ולכן הוא <u>לא</u> מגדיל את מרחב החיפוש בהתקפת מילון על משתמש

ספציפי. הוא רק מקשה על חישוב מראש.)

מס' מחברת: _ _ _

שאלה 5 (10 נק')

למכונת פופקורן דיגיטלית יש ממשק רשת עם כתובת IP קבועה (10.7.7.7) . המשתמש יכול אז לפתוח את האתר בצורה "http://10.7.7.7/control" כדי לגשת לממשק ההפעלה של המכונה ולתת פקודות. הממשק מורכב מדף HTML ובו קישורים בשם "הדלק", "כבה", "הוסף מלח" וכו' שיוצרות פניות כמו `http://10.7.7.7/control?cmd=AddSalt`. הקלקה על קישור גורמת לביצוע הפעולה. הניחו כי אין אפשרות ל- control hijacking בקוד של מכונת הפופקורן.

א. [5 נק'] סגל הקורס רכש מכונת פופקורן כזו. כיצד תגרום לקילקול הפופקורן שלהם בשל תוספת מלח מוגזמת, באמצעות דוא"ל בלבד? (ניתן להניח שממשק ההפעלה במכונת הפופקורן מומש באופן נאיבי, ושהדוא"ל נפתח ע"י המתרגל המפעיל את המכונה, אבל הוא לא לוחץ על שום לינק בתוך האימייל.) הדואל יכול להיות HTML, ותוכנת הדואל תציג את כולו.

התקפת Cross-Site Request Forgery . המתקיף ישלח דואל HTML המכיל (לדוגמה) העלאת

תמונה מכתובת ה-HTTP שבשאלה: <img src="http://10.7.7.7/control?cmd=AddSalt">

תוכנת הדואל תיצור בקשת HTTP אל המכונה, והמכונה תציית.

(הערה: זו לא התקפת XSS. דואל HTML אמור לכלול תמונות וכו', לכן אין כאן בעיית סניטציה.)

ב. [5 נק'] הצע שינוי קטן לתוכנת מכונת הפופקורן המונע את ההתקפה מסעיף א'. (על הממשק להישאר מבוסס דפדפן).

כמה אפשרויות:

---

* anti-CSRF token : תוספת פרמטר אקראי, העובר ב-URL (או בשדה טופס ב-POST, אבל

לא ב-cookie, זה לא יעזור כי הדפדפן תמיד מצרף cookies של דומיין). הקישורים בדף הממשק

יכללו את הפרמטר, ובעת בקשת פעולה הוא יבדק.

---

* המכונה תבדוק את ה- referer כשהי מקבלת בקשת control.

---

* תוספת הזדהות למכונה (לדוגמה, סיסמה). כדי למנוע CSRF חשוב שהבדיקה תהיה ברגע

האחרון (באותה בקשת HTTP שמבצעת את ההמלחה), ולא לפני כן (לדוגמה בכניסה לתפריט).

שאלה 6 (12 נקודות)

מערכת הסליקה של בנק מבוססת על קובץ העברות הבנוי מסדרה של רשומות. כל רשומה בקובץ מתארת העברה כספית אחת, היא בת 16 בתים ונראית כך:

Bytes 0-3	Bytes 4-7	Bytes 8-11	Bytes 12-15
מספר חשבון לחיוב	מספר חשבון לזיכוי	סכום להעברה	קוד-ביקורת

קוד הביקורת מחושב בנפרד לכל רשומה, כפונקציה של שאר השדות ברשומה. מחשב הבנק לא יבצע העברה אם הקוד-ביקורת באותה רשומה לא מתאים לשאר השדות.

א. [4 נק'] מר איקס הוא פקיד מושחת בבנק שיכול לעדכן את קובץ ההעברות (לשנות, להוסיף, או למחוק רשומות כרצונו), אבל הוא איננו יודע איך לייצר קודי ביקורת. נניח כי מר איקס רואה בקובץ ההעברות רשומה שמזכה את חשבונו הפרטי ב-100 שח. הציעו שיטה בה מר איקס יוכל לגרום למחשב הבנק להעביר אל חשבונו הפרטי סכומים גדולים כרצונו.

הוא יכול לשכפל את הרשומה שהוא מכיר מספר פעמים כרצונו, קוד הביקורת תקין.

(תשובה שאפשר לבצע fuzzing ולנסות לקלוע לקוד ביקורת מתאים עבור העברה אחרת

גרמה להורדת נקודות כי סיכוי ההצלחה $1/2^{32}$ כלומר לוקח הרבה זמן ומצריך המון שטח

דיסק – התוקף צריך שהבנק יבדוק את קוד הביקורת.)

מס' מחברת: _ _ _

ב. [4 נק'] הדליפו למר איקס כי קוד הביקורת לרשומה (FromAccount, ToAccount, XferAmount) מיוצר בצורה הבאה:

```
Code = BankSecret;
Code = Rotate5(Code ^ FromAccount);
Code = Rotate5(Code ^ ToAccount);
Code = Rotate5(Code ^ XferAmmount);
```

כאשר Rotate5 מבצעת גלגול (מעגלי) של 5 ביטים שמאלה וסימן "^" הוא bitwise xor. הערך הסודי **BankSecret** לא הודלף. נניח כי מר איקס שוב רואה בקובץ ההעברות רשומה שמזכה את חשבונו הפרטי ב-100 שח. תארו דרך שבה הוא יכול לשנות את הרשומה הזו (בלבד) כך שמחשב הבנק יעביר לחשבונו 1,000,000 שח .

מס תשובות: 1, כל הפעולות הפיכות אז אפשר לגלגל את הקוד אחורה ולגלות את סוד הבנק.

2, אפשר לגלגל אחורה רק את הפעולה האחרונה, לקסר החוצה את הסכום הישן ולקסר

פנימה את החדש, ולגלגל קדימה לקבלת קוד תקין. 3, קסור הוא ליניארי , לכן אפשר לחשב

את הקסור בין הסכום הישן והחדש, מגולגל למקום המתאים, ולקסר את התוצאה עם קוד

הביקורת הקיים.

ג. [4 נק'] הציעו שינויים במבנה קובץ ההעברות ובתוכנת הבנק שיחסמו את החולשות שמר איקס ניצל בסעיפים א ו-ב.

נגד ב: חתימה דיגיטלית או שימוש ב-hash קריפטוגרפי סטנדרטי כמו Sha1.

נגד א: למספר העברות במספור גלובאלי ולוודא שהוא לא חוזר על עצמו.

(חתימה אחת על כל הקובץ – ניקוד חלקי, כי עדיין אפשר לשכפל קבצים שלמים.)

(קוד רנדומי בתוך הרשומה כדי לשמור על טריות – ניקוד חלקי, כי הצד הבודק לא יודע למה

לצפות והתוקף יכול אולי לשים שם ערך כרצונו, וחוץ מזה כדי לחסום שיכפול הצד הבודק

צריך לשמור את כל הערכים שהיו בשימוש עד כה כלומר יש כאן דרישת זיכרון קשה למימוש.)