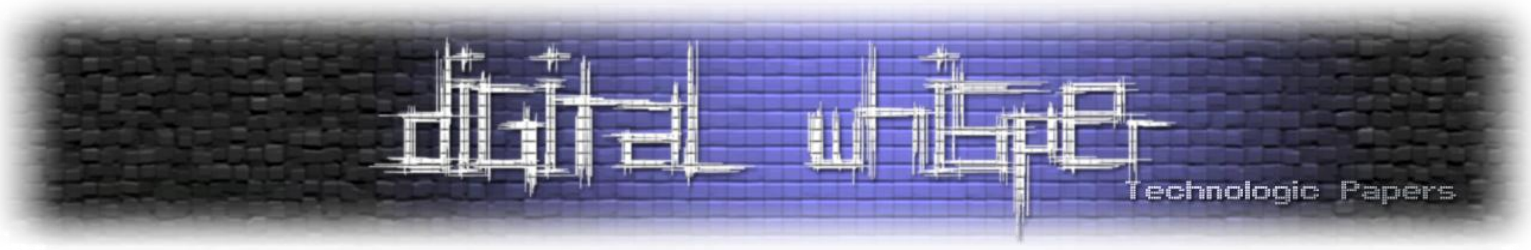


Hacking Kindle DRM

מאת LaBBa

את המאמר נפתח בפירוש מושג ה-DRM או בעברית-נז"ק. פירוש המושג Digital Rights Management, או "ניהול זכויות דיגיטלי", הוא פשוט למדי - החברה המשתמשת במושג בוחרת להגביל את המשתמשים באמצעות טכניקות שונות. טכניקות אלו יכולות להיות קשורות בתוכנה או בחומרה ומטרתן להבטיח כי הצרכן של אותו המוצר לא יוכל להעביר את תוכן המוצר לאדם אחר.

עצם העיקרון הכללי, המגביל את לקוחות החברה בכל הנוגע להעברת תוכן, גורר אחריו תוצאות ותופעות לוואי בלתי נמנעות. כלומר, אדם הרוכש מוצר כלשהו יתקשה, או לא יוכל להעביר את התוכן למקום אחר או לשימוש בתוכנה אחרת. לדוגמא, נניח כי חברה משתמשת ב-DRM על מנת להגן על שיר בצורה מסוימת, הגנה זו תמנע האזנה לשיר בכל נגן שאינו מסופק על ידי אותה החברה. אם ברשות הלקוח נגן אחר, הוא לא יוכל להאזין לשום שיר שרכש מפני שמדובר בקובץ יוצא דופן ומוגן. בעשור האחרון, חוקקה ממשלת ארצות הברית את חוק המילניום, האוסר על לקוחות ומשתמשים להסיר את הגנות ה-DRM, משום שעצם ההסרה היא פגיעה בזכויות יוצרים. ואכן, גם פה בארץ היו תיקונים לחוק זכויות היוצרים. תיקונים אלו נקבעו בשנת 2007 ומציגים אפשרות מעט שפויה יותר- החוק מתיר לאדם אשר רכש מוצר להסיר את ההגנות של זכויות היוצרים על מנת להתאימו לתוכנה אחרת, כמו גם יצירת גיבוי של המוצר והקלות שונות שמותרות אך ורק לבעל המוצר, אך בשום פנים ואופן אין למכור או להעביר את התוכן לאחר הסרת ההגנה.

Kindle

הקינדל יצא לראשונה לשוק כחומרה המאפשרת למשתמשים לרכוש ספרים מ-Amazon ולקרוא בהם על גבי אותו המוצר. כצפוי, חברת Amazon בחרה להגן על הספרים שלה בשיטות הצפנה ייחודיות כך שלא יהיה ניתן להעביר ספרים מאדם לאדם וכן שלא יתאפשר לקרוא את אותם הספרים על אף מוצר מלבד אותו מוצר יחיד שנרכש עבור הספר. הגנות אלו הקשו על המשתמשים, שכן לאחר רכישת הספרים לא יכלו לקרוא בהם למעט על גבי אותה חומרה ספציפית שיועדה לכך. כמו כן, לא התאפשר למשתמשים להמיר את הקבצים לפורמט PDF או פורמטים סטנדרטיים אחרים המיועדים לקריאה במחשבים ביתיים.

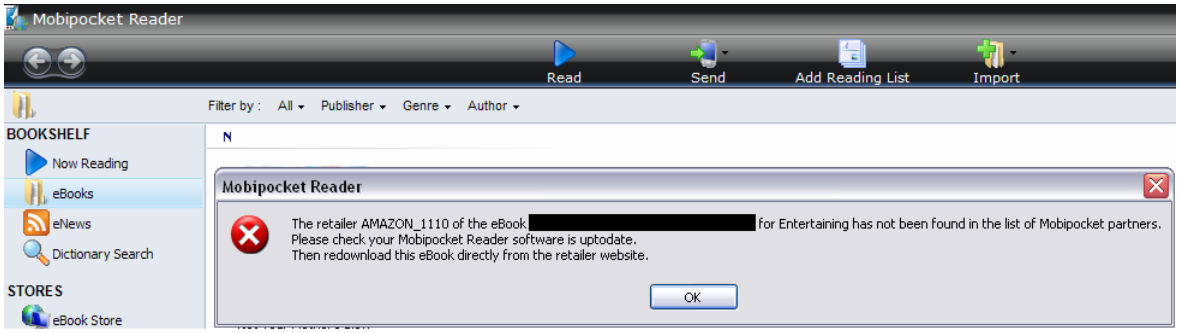
Amazon בחרה לתמוך אך ורק בפורמטים AZW, MOBI, ו-TPZ ואף לא תמכה בקריאת ספרים שאינם מוגנים בהתאם לשיטה שקבעה, כך שלא ניתן היה להשתמש במכשיר זה לכל ספר אחר.

בתגובה למחדל, קהילת ההאקרים החלה לנסות וללמוד את הטכנולוגיה העומדת מאחורי Kindle על מנת להבין כיצד ניתן יהיה להסיר את ההגנה ולאפשר קריאה של הספרים לא רק בעזרת החומרה, כמו גם לאפשר ל-Kindle לקרוא תוכן נוסף שלא בהכרח נרכש דרכם ולתמוך בפורמטים נוספים מלבד אלו שהוגדרו כתקניים על ידי Amazon. ההאקר הראשון שהצליח במשימה היה איגור סקוצ'ינסקי (Igor Skochinsky), שמצא את המספר הייחודי של כל מוצר (UUID), את האלגוריתם אשר מספק מהמספר הייחודי (ופרמטרים נוספים על המשתמש של המוצר) מפתח ייחוד שניקרא בשם PID (platform ID). לאחר מכן הוא קודד תוכנה המאפשרת למשתמש ליישם את החתימה של Amazon על גבי קבצים שאינם מוגנים, כך שניתן יהיה להשתמש בחומרה על מנת לקרוא ספרים שלא נרכשו דרך החברה. בנוסף, מצא סקוצ'ינסקי כי Amazon משתמשת בשיטת הצפנה הנקראת PC1 או בשמה המלא Pukall (הצפנה זו נקראת על שם אלכסנדר פוקאול (Alexander Pukall)).

זמן קצר לאחר תגליות אלו, האקר בשם DarkReverser עבד על הסרת ההגנה של ה-DRM כך שניתן יהיה להסיר את ההגנות ולקרוא את הספרים על ידי תוכנות אחרות מלבד התקן החומרה Kindle ואף ביצע עדכונים ותחזוקה בקוד של איגור סקוצ'ינסקי. הקוד של DarkReverser אפשר למשתמש, בהנתן ה-PID של המוצר, להסיר את ההגנות החלות עליו ולקבל קובץ סופי של ספר שאינו מוגן.

Kindle For PC

שלוש שנים לאחר מכן, החליטה Amazon כי היא רוצה להתפתח והחלה להציע ללקוחותיה מוצר חדש המאפשר קריאת ספרים מוגנים גם על גבי מחשבים ביתיים. הבעיה העיקרית בתוכנה זו היא שמדובר בגרסאת Beta, גרסא שנכון לעכשיו לא מכילה מאפיינים מתקדמים. בנוסף, כיום קיימות תוכנות המסוגלות לקרוא ספרים בפורמט MOBI ש-Amazon תומכת בהן ויש להן יכולות מיוחדות, כגון הדגשות בתוך הטקסט, הערות של המשתמש וכן הלאה. אך מה יקרה אם נרצה לפתוח ספר שרכשנו המכיל הגנות DRM של Amazon?



ובכן, כאן אני נכנס לסיפור. ההנחה הראשית שיצאתי ממנה היא Amazon לא תשבור את ממשק האבטחה שבנתה ל-Kindle ותמשיך עם אותה הטכנולוגיה. בהתבסס על כל העובדות שנצברו, התחלתי לחשוב כיצד אוכל למצוא את ה-PID שנוצר על המחשב האישי לעומת מה שקודמי עשו עם מוצר החומרה ה-Kindle. המחשבה הראשונה שעולה לכל אחד, הניגש למשימה דומה, היא כנראה "איפה לעזאזל מתחילים?" והתשובה במקרה זה היא פשוטה מכפי שתוכלו לדמיין. לקחתי את קוד המקור של DarkReverser והתמקדתי בחלק האחראי להסרת ההגנה, לאחר שכבר גילינו את ה-PID.

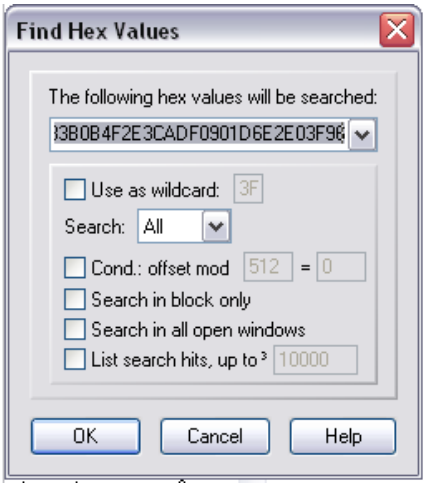
הנה חלק מהקוד של DarkReverser:

```
def parseDRM(self, data, count, pid):
    pid = pid.ljust(16,'\0')
    keyvec1 = "\x72\x38\x33\xB0\xB4\xF2\xE3\xCA\xDF\x09\x01\xD6\xE2\xE0\x3F\x96"
    temp_key = PC1(keyvec1, pid, False)
```

כפי שניתן לראות, הוא קורא לפונקצית ההצפנה PC1 עם 3 ארגומנטים, כאשר המעניין מביניהם הוא דווקא לא ה-PID אלא ה-keyvec1. משתנה זה הוא מעניין יותר עבורנו מפני שהוא ניראה כרצף בתים קבוע הנדרש על מנת להצליח לפרוק הצפנה של קבצים, בנוסף ל-PID.

ברגע שראיתי את הווקטור הקבוע המדובר, פתחתי HexEditor לחפש את רצף הבתים הזה:

```
007FB130  74 39 C8 00 05 00 00 00  03 00 00 00 E8 D9 C7 00
007FB140  06 00 00 00 04 00 00 00  D4 6F C8 00 07 00 00 00
007FB150  05 00 00 00 C8 6F C8 00  05 00 00 00 06 00 00 00
007FB160  74 00 72 00 75 00 65 00  66 00 61 00 6C 00 73 00
007FB170  65 00 75 00 6E 00 64 00  65 00 66 00 69 00 6E 00
007FB180  65 00 64 00 6E 00 75 00  6C 00 6C 00 30 00 4E 00
007FB190  61 00 4E 00 49 00 6E 00  66 00 69 00 6E 00 69 00
007FB1A0  74 00 79 00 00 00 00 00  72 38 33 B0 B4 F2 E3 CA
007FB1B0  DF 09 01 D6 E2 E0 3F 96  FF FF FF FF FF FF FF FF
007FB1C0  00 00 00 00 00 00 00 00  00 00 00 00 22 00 01 00
007FB1D0  22 00 00 00 00 00 00 00  26 00 01 00 26 00 00 00
007FB1E0  00 00 00 00 27 00 01 00  27 00 00 00 00 00 00 00
007FB1F0  3C 00 01 00 3C 00 00 00  00 00 00 00 3E 00 01 00
007FB200  3E 00 00 00 00 00 00 00  A0 00 01 00 A0 00 00 00
007FB210  00 00 00 00 A1 00 01 00  A1 00 00 00 00 00 00 00
```



כפי שניתן להבין מתרשים זה, רצף הבתים הזה קיים בשלמותו בתוך קובץ ההרצה של תוכנת ה- Kindle For PC. די נדהמתי לגלות כי הרצף קיים גם בתוכנת המחשב ורציתי לדעת היכן בקוד של התוכנה נעשה שימוש בערך זה, מפני שאם אמצא את המקום שמשתמש בערך הזה, שם אמור להמצא גם ה-PID (על פי דוגמת הקוד של DarkReverser).

על מנת למצוא את המקום שבו התוכנה תשתמש בערך זה, נעזרתי ב-IDA:

```
:00BFC5A8 dword_BFC5A8    dd 0B0333872h
:00BFC5A8
:00BFC5AC dword_BFC5AC    dd 0CAE3F2B4h
:00BFC5B0 dword_BFC5B0    dd 0D60109DFh
:00BFC5B4 dword_BFC5B4    dd 963FE0E2h
:00BFC5B8                 db 0FFh
:00BFC5B9                 db 0FFh
```

```
; DATA XREF: sub_5708C0+EA↑r
; sub_5708C0+1B8↑o
; DATA XREF: sub_5708C0+F0↑r
; DATA XREF: sub_5708C0+F6↑r
; DATA XREF: sub_5708C0+FF↑r
```

References

כפי שאתם רואים, IDA מספקת את ה-references להיכן בקוד משתמשים בערך זה:

```
:005709AA                    mov     ecx, ds:dword_BFC5A8
:005709B0                    mov     edx, ds:dword_BFC5AC
:005709B6                    mov     eax, ds:dword_BFC5B0
:005709BB                    mov     [esp+0C4h+var_AC], ecx
:005709BF                    mov     ecx, ds:dword_BFC5B4
```

וכפי שניתן להבחין, בכתובת: 5709AA מתחילה השמת הערך של הווקטור למשתנים לוקאליים של הפונקציה. לאחר מכן, מה שנותר רק לעשות הוא להפעיל את ה-Debugger העדיף עלינו, לשים שם נקודת עצירה ולנסות לפתוח את הספר המוגן שקנינו. למרות שללא ספק ניתן לבצע Debug עם IDA, אני מעדיף לבצע תהליך זה עם OllyDbg.

וכך, אם נשים נקודת עצירה בנקודה הזו OllyDbg יעצור:

```
005709A6  . 837D 04 02      CMP DWORD PTR SS:[EBP+4],2
005709AA  . 8B0D A8C5BF00   MOV ECX,DWORD PTR DS:[BFC5A8]
005709B0  . 8B15 ACC5BF00   MOV EDX,DWORD PTR DS:[BFC5AC]
005709B6  . A1 B0C5BF00     MOV EAX,DWORD PTR DS:[BFC5B0]
005709BB  . 894C24 18       MOV DWORD PTR SS:[ESP+18],ECX
005709BF  . 8B0D B4C5BF00   MOV ECX,DWORD PTR DS:[BFC5B4]
```

אם תשימו לב, בצד תוכלו לראות כי יש כאן לולאות מאוד גדולות בקוד (כיאה לקוד של אלגוריתמים) ולכן לא ממש התעניינתי בניתוח הקוד, אלא רק רציתי לראות האם משהו יצוץ לי מול העיניים כאשר אלחץ על F8 (step over). לאחר כ-200 לחיצות אני מגיע לשורת הקוד בכתובת: 570A94

```
00570A75  . 83C4 0C         ADD ESP,0C
00570A78  . B9 A8C5BF00     MOV ECX,Patched.00BFC5A8
00570A7D  . 8D9424 A400000  LEA EDX,DWORD PTR SS:[ESP+A4]
00570A84  . C78424 A000000  MOV DWORD PTR SS:[ESP+A0],Patched.00C
00570A8F  . E8 FCECFDFF     CALL Patched.0054F790
00570A94  . 8D4424 18       LEA EAX,DWORD PTR SS:[ESP+18]
00570A98  . 50              PUSH EAX
00570A99  . B8 10000000     MOV EAX,10
00570A9E  . 8D4C24 1C       LEA ECX,DWORD PTR SS:[ESP+1C]
00570AA2  . 8BF2            MOV ESI,EDX
00570AA4  . E8 37EDFDFF     CALL Patched.0054F7E0
Stack address=0013F538, (ASCII "-31[redacted]")
EAX=0013F538, (ASCII "-31[redacted]")
```

כאן תוכלו להבחין כי זו אכן כן מחרוזת. בהתאם, ביטלתי את נקודת העצירה הקודמת, שמתי נקודת עצירה חדשה על השורה הזאת (כפי שציינתי יש כאן הרבה לולאות) ולאחר מכן לחצתי על F9 (run).

לאחר מכן קיבלתי מחרוזת חדשה, הדבר חזר על עצמו פעמיים ולאחר מכן התוכנה המשיכה לרוץ ללא מפריע והציגה את הספר שרכשתי.

המחרוזת האחרונה שהוצגה הייתה בת 8 תווים, אם זה אכן ה-PID אז כל מה שנותר לעשות הוא לקחת את התוכנה ש-DarkReverser כתב ולנסות להסיר את ההגנה על ידי שימוש ב-PID שמצאתי.

מהלך זה הביא להפתעה לא נעימה:

```
C:\MobiDeDRM>mobidedrm.py enc.prc out.prc 2       E
MobiDeDrm v0.07. Copyright (c) 2008 The Dark Reverser
Error: invalid PID checksum
```

לאחר עיון ובדיקה שניה של הדוגמאות שהובאו על ידי חוקרים אחרים, גיליתי באחד ההסברים פיסה של מידע שימושי, גודל ה-PID צריך להיות 10 תווים! משמעות הדבר היא שחסרים לנו שני תווים וכי תווים אלו הם רק עבור בדיקת חוקיות. נקרא את הקוד שמזהה לנו מהו גורם ל-Bad CheckSum:

```
if checksumPid(pid[0:-2]) != pid:
    raise DrmException("invalid PID checksum")
```

כלומר, אם אשלח לפונקציה רק 8 תווים, במקום 10 (זו המשמעות של pid[0:-2]), אני אמור לקבל אותה מחרוזת בעלת 10 תווים שאליה אני צריך להשוות את המחרוזת שלי. משמעות הדבר היא שפונקצית ה-checksumPid ש-DarkReverser מימש מקבלת 8 תווים ומחזירה לי 10 תווים -את התווים החסרים.

בהתאם, לקחתי מהקוד של DarkReverser רק את הפונקציה של ה-cehcksumPid ויצרתי קובץ חדש כך:

```
import sys,struct,binascii

def checksumPid(s):
    letters = "ABCDEFGHIJKLMNPQRSTUVWXYZ123456789"
    crc = (~binascii.crc32(s,-1))&0xFFFFFFFF
    crc = crc ^ (crc >> 16)
    res = s
    l = len(letters)
    for i in (0,1):
        b = crc & 0xff
        pos = (b // l) ^ (b % l)
        res += letters[pos%l]
        crc >>= 8
    return res

print "PID Fixer"
pid = sys.argv[1]
print "the fixed PID is: " + checksumPid(pid)
```

אם ננסה בשלב זה לשלוח את 8 התווים שלנו לסקריפט החדש , זאת תהיה התוצאה:

```
C:\MobiDeDRM>CalcFixed.py 2[redacted]E
PID Fixer
the fixed PID is: 2[redacted]EHR
```

שלחנו 8 תווים וקיבלנו 10 תווים! עכשיו ננסה שוב להשתמש בסקריפט של DarkReverser, הפעם עם 10 תווים.

```
C:\MobiDeDRM>mobidedrm.py enc.prc out.prc 2[redacted]HR
MobiDeDrm v0.07. Copyright (c) 2008 The Dark Reverser
Decrypting. Please wait... done
```

אם ננסה לפתוח את הספר עם תוכנה אחרת כגון Mobipocket Reader לא נקבל יותר את הודעת השגיאה ונוכל להשתמש בפונקציות היותר מתקדמות מאשר בגירסת הבטא.

סיכום

לסיכום, עלינו לזכור כי בסופו של דבר ישנו חוק מנחה אחד- אם זה מסוגל לרוץ בצורה מלאה פעם אחת, תמיד יהיה אפשר לפרוץ את המוצר. כל עוד כל קטעי הקוד הורצו בצורה לוקאלית על המעבד שלנו אין ל-Amazon הרבה ברירות והם יודעים את זה. מאחר והם חייבים לספק פיתרון אשר יאפשר למשתמש לקרוא את הספרים גם בצורה שאינה מחייבת קישור אינטרנט, הם חייבים להריץ את כל הקוד על המעבד שבמחשב שלנו. מסיבה זו בדיוק, Amazon תמיד יפסידו במשחק הזה היות והארכיטקטורה של x86 ביסודה מאפשרת לעשות הכל ללא הגבלה- אם החומרה לא מגבילה אותנו אז התוכנה פרוצה.

הערה: כיום עדיין לא אפשרי לפתוח ספרים מסוג TPZ בעזרת שום תוכנה אחרת מאחר וזהו פורמט לא סטנדרטי כמו MOBI אלא פורמט פרטי של Amazon. יש צורך להשתמש בתוכנות אחרות אשר ימירו את הפורמט הפרטי של Amazon לפורמט HTML משם אפשר להמיר לכל פורמט שנרצה כדי לפתוח בכל תוכנה שנרצה. לאחרונה, פורסם בפורום שלDarkReverser כלים שיודעים להסיר את ה-DRM מקבצי TPZולהמיר אותם לקבצי HTML להמרה עתידית לכל סוג של פורמט שנרצה.

בזמן כתיבת שורות אלו, הגירסא האחרונה היא: 1.6: http://www.box.net/shared/gqukrbp0js

כמובן- השימוש במידע זה הוא לצורכי לימוד בלבד ולא למטרות פיראטיות!

מקורות וקישורים

- DarkReverser home page -http://darkreverser.wordpress.com/
- DarkReverser script -http://pastebin.com/f7be270a9
- Igor Skochinsky -http://igorsk.blogspot.com/