

Digital Whisper

גליון 73, יוני 2016

מערכת המגזין:

אפיק קסטיאל, ניר אדר

מייסדים:

אפיק קסטיאל

מוביל הפרויקט:

אפיק קסטיאל, ניר אדר

עורכים:

D4d, תומר זית, 0x3d5157636b525761 ואופיר בק

כתבים:

יש לראות בכל האמור במגזין Digital Whisper מידע כללי בלבד. כל פעולה שנעשית על פי המידע והפרטים האמורים במגזין Digital Whisper הינה על אחריות הקורא בלבד. בשום מקרה בעלי Digital Whisper ו/או הכותבים השונים אינם אחראים בשום צורה ואופן לתוצאות השימוש במידע המובא במגזין. עשיית שימוש במידע המובא במגזין הינה על אחריותו של הקורא בלבד.

פניות, תגובות, כתבות וכל הערה אחרת - נא לשלוח אל editor@digitalwhisper.co.il

דבר העורכים

מי שאיכשהו הצליח לפספס ולא שם לב, במהלך חודש מאי פורסם [הגליון ה-69](#) של המגזין האגדי - Phrack. מי מביניכם שמכיר אותי, יודע שיש לי פינה מיוחדת בלב למגזין הנ"ל, למה שהוא מסמל ולכל מה שמסביבו. ואף יותר מזה - אם אתם מהאלה ששמים לב לפרטים הקטנים יהיה לכם קשה מאוד לפספס שגם Digital Whisper הושפע מהנ"ל לא מעט.

קשה להגיד שהמעצר של Knight Lightning השפיע עלי (בכל זאת, הייתי אז בן 3...), וקשה להגיד שהרגשתי את ההבדל כאשר The Circle of Lost Hackers החלו לקחת את הפיקוד על המגזין. לא הייתי שם בגליון ה-49 כאשר [Aleph One הרעיד את הרשת](#) או בגליון ה-51 כאשר [Fyodor הכריז על nmap](#).

אך עם זאת, Phrack מבחינתי מסמל את אחד הדברים היפים ביותר שיש היום (מעבר לזה שיש לי חולשה ל-ASCII Art), בשבילי Phrack הוא איזשהו טלפורטר לסצינת ההאקינג כאשר היא עוד רק החלה. החברה שם מצליחים לשמר כמעט בכל אספקט את אותו דבר יפה שהיה כאן לפני שסצינת ההאקינג "התמסחרה", לפני שזה היה באופנה לכנות את עצמך "האקר", לפני שבייתו את המילה הזאת, ולפני שהיה כזה מקצוע.

כיום ניתן לראות כל כך הרבה חבר'ה שמתראיינים בתקשורת ותחתיים כתוב "האקר" או "האקר לשעבר"... איך לעזאזל אפשר להיות "האקר לשעבר"?! בחיפוש קצרצר בגוגל ניתן למצוא שיש כל כך הרבה הגדרות ותפקידים וקורסים והכשרות ובתי ספר שרק אומרים לך "בוא, תלמד ותוך חצי שנה של לימודי ערב אתה תהיה האקר ויהיה לך מקצוע של האקר", אני לא יודע למה, אבל אותי זה מעציב.

בעצם, אני חושב שאני כן יודע למה: כפי שאני (ואני בטוח שעוד רבים) תופס את העניין, האקר זה לא מקצוע, אי-אפשר לשבת בכיתה וללמוד להיות האקר, האקר זה דרך חיים, זאת תפיסה שהיא מעבר ללוח וגיר, היא מעבר למקצוע והיא גם מעבר ליכולת כזו או אחרת. להיות האקר זה דרך חיים שלמה. זה טבוע בבן אדם וזה מורגש כמעט בכל צעד ופעולה שהוא עושה, לכן גם אי-אפשר להיות "האקר לשעבר", אפשר להיות אדם שפעם עסק באבטחת מידע, או להיות מורה לשעבר, אבל אי-אפשר להיות האקר לשעבר.

ולכן, בכל פעם שגליון נוסף של Phrack מתפרסם, מבחינתי זה יום חג, אני מחכה שיהיה לי ערב פנוי או אפילו מחכה לסופ"ש הקרוב, יושב וקורא את דברי הפתיחה בשקיקה, קורא את Loopback במידה ומתפרסם, חוזר וקורא את דברי הפתיחה של גליונות קודמים, מדי פעם אני חוזר וקורא את המסמך האגדי של Loyd Blankenship (בפעם המליון בערך), מתחבר ל-BBS-ים שעדיין בחיים, קורא את החדשות ואת כל מה שמסביב - ולאחר מכן, רק בסוף, מתפנה לקרוא גם את התוכן.



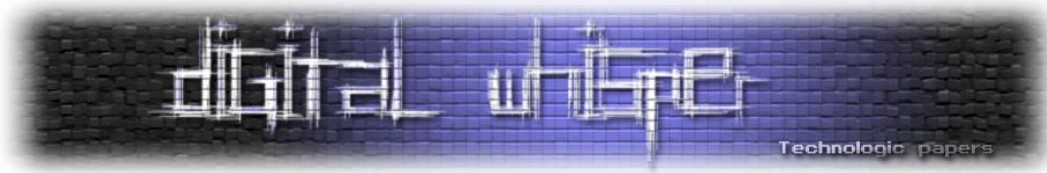
עכשיו, שלא תבינו אותי לא נכון - התוכן הוא הסיבה העיקרית למגזין, ובלעדיו אפשר לומר שלמגזין אין זכות קיום, אבל אם התוכן מגיע (או מתקבל אצל הקורא) ללא כל הרקע שמסביב - זה לא יהיה אותו הדבר בכלל. זה נכון שלכמעט כל מאמר שמופיע ב-Phrack אין שני, אבל אם תשאלו אותי - לא הייתי רוצה אפילו פי 2 מכמות המאמרים אם היא הייתה מגיעה ללא כל מה שמסביב.

ואולי זה רק אני, רומנטיקן שעבר זמנו...

וכמובן, לפני הכל - תודות לכל מי שתרום החודש למגזין, השקיע מזמנו, ישב וכתב על מנת שלכולנו יהיה טעם להמשיך לחיות: תודה ל-D4d, תודה לתומר זית, תודה ל-0x3d5157636b525761 ותודה רבה לאופיר בק!

קריאה מהנה!

ניר אדר ואפיק קסטיאל.



תוכן עניינים

2	דבר העורכים
4	תוכן עניינים
5	פתרון אתגר המוסד 2016
27	משטחי תקיפה באפליקציות Android - חלק א'
38	מבוא לאסמבלי
49	דברי סיכום

פתרון אתגר המוסד 2016

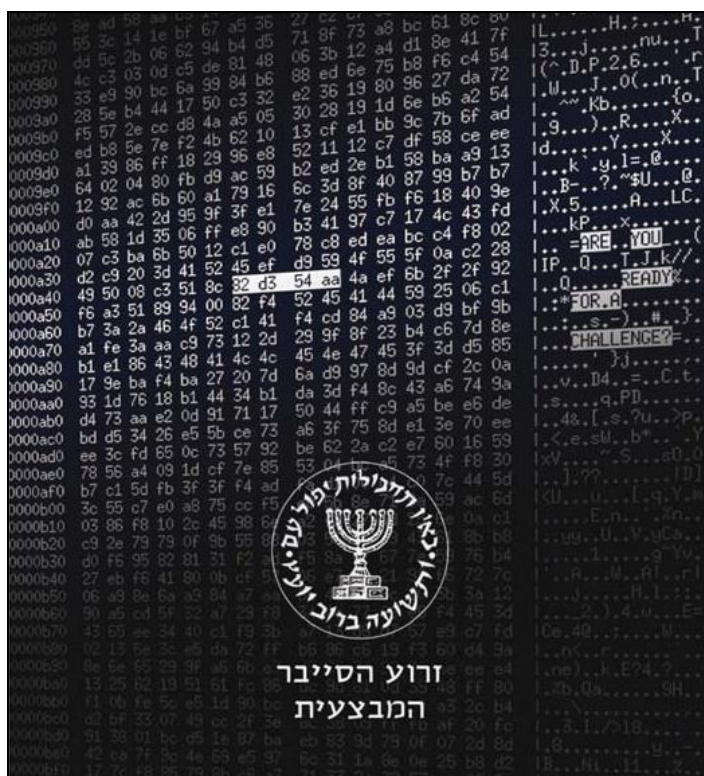
מאת D4d ותומר זית

הקדמה

ביום העצמאות האחרון, "זרוע הסייבר המבצעית" של המוסד הישראלי פרסם אתגר האקינג למטרת איתור וגיוס מועמדים פוטנציאליים לשורותיו. D4d ואני (תומר זית) פתרנו את האתגר במקביל (כל אחד בנפרד), ולאחר שסיימנו אותו - החלטנו להשוות דרכי פתרון וכך נולד מאמר זה. האתגר הורכב ממספר שלבים, בכל שלב היה נדרש ידע והבנה במספר לא קטן של נושאים. על מנת להגיע להסבר המדויק ביותר החלטנו לחלק את העבודה בינינו וכל אחד כתב על שלבי הפתרון הקרובים לעיסוקו היום-יומי. לאחר כשבוע האתגר הסתיים ולכן ראינו לנכון לפרסם מאמר זה.

שלב מקדים - למצוא את הדרך לאתגר.

חלקנו שמעו על האתגר דרך פרסום בפייסבוק, חלקנו ראו את הפרסומת בעיתון, אחרים ראו פרסומים באתרים שונים, אך בסופו של דבר, הכל הוביל לתמונה הבאה:



פתרון אתגר המוסד 2016
www.DigitalWhisler.co.il



כפי שניתן לראות, בתמונה מודגשים ברקע לבן מספרים הקסה דצימליים, המספרים הנ"ל מתורגמים בסופו של דבר לכתובת האתר שבו מאוכסן האתגר. ניתן לתרגם את המספרים ההקסה דצימליים לכתובת IP בכמה דרכים שונות. נשתמש ב-Python על מנת לעשות זאת בדרך אלגנטית, חשוב להבין שכל מספר הקסה דצימלי מתורגם לאוקטטה בכתובת ה-IP של האתר:

```
import socket
print socket.inet_ntoa('\x82\xd3\x54\xaa')
```

דרך נוספת:

```
print "%d.%d.%d.%d" % (0x82, 0xd3, 0x54, 0xaa)
```

התוצאה: 130.211.84.170.

שלב ראשון - XModem-CRC

הסבר על המשימה:

Challenge #1

Good morning Agent C!

We require your skills for an urgent search & rescue mission.

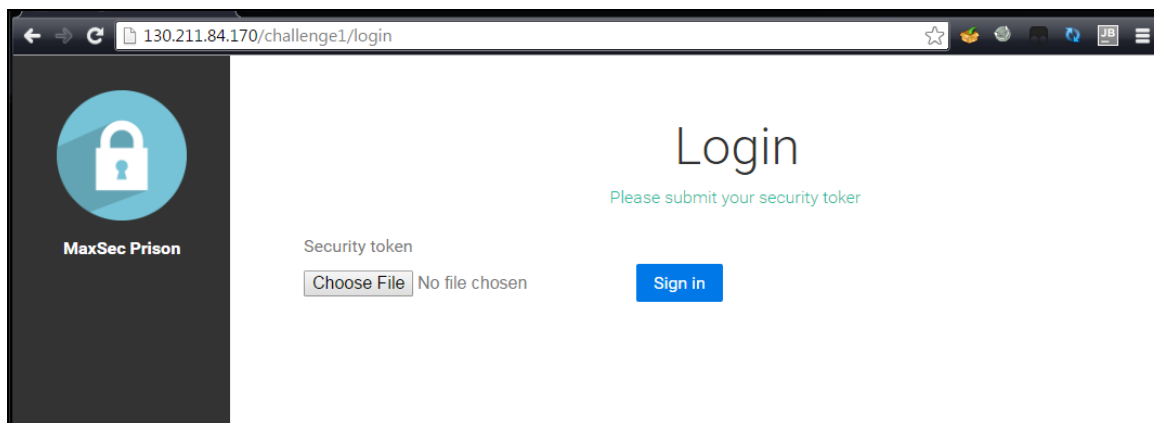
One of your colleagues has been taken hostage by an unidentified group, and is being held in a previously unknown holding facility. Our SIGINT squad has been successful in geo-locating the facility and found indications of an electronic lock mechanism in the main entrance.

The rescue team needs your help in opening this mechanism so they can enter and search the premises.

Good luck,
A.

[Start Challenge](#)

דף ה-Login:



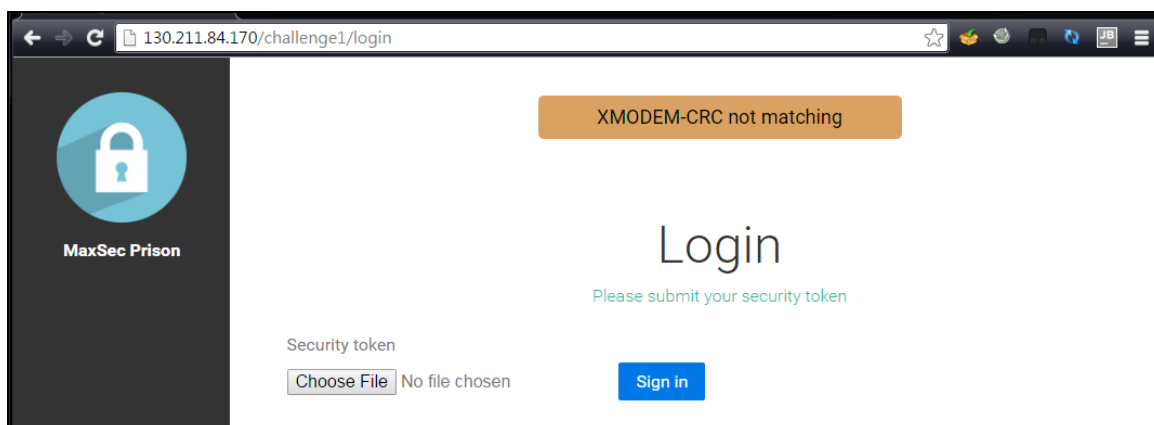
המטרה: להתחבר בהצלחה למערכת.

הדרך: צריך להעלות קובץ המהווה Token הזדהות למערכת.

מה אפשר להעלות למערכת?:

בעזרת משחק קצר של ניסיון-וטעייה ניתן להבין לא מעט מאפיינים של הקובץ אותו יש להעלות: במידה והיינו מעלים קובץ גדול מדי - היינו מקבלים שגיאה שהקובץ גדול מדי, במידה והיינו מעלים קובץ קטן מדי - היינו מקבלים שגיאה כי הקובץ קטן מדי. במידה והגודל היה מתאים אך הקובץ לא התחיל עם Header של קובץ PNG - היינו מקבלים שגיאה בהתאם.

השגיאה שאיתה נתקלנו הייתה שגיאה כי הקובץ אכן תמונה ואכן בגודל המתאים (בערך בין 7kb ל-14kb) והקובץ אכן תמונה, אך הוא אינו בעל CRC ספציפי:



כפי שניתן לראות קיבלנו שגיאה: XMODEM-CRC not matching, [אחרי שנקרא על XMODEM-CRC](#) נבין שמדובר ב-CRC16. בשלב זה אנחנו מבינים מה עלינו לעשות - אבל לא איך לעשות זאת.

קוד המקור:

לאחר שמביטים בקוד המקור של דף ה-Login ניתן לראות שהקישור של תמונת הלוגו לא שגרתית, במקום קישור עם סיומת של תמונה יש לנו קישור לדף עם פרמטרים (h ו-multiple), הקישור נראה כך:

<http://130.211.84.170/challenge1/get-image?name=logo.png&h=87d41d15f&multiple=0>

משמעות הפרמטרים:

- **multiple** - מרמז על תוכן מרובה ונראה כי הוא משתנה בולאני.
- **h** - מזכיר האש אך קצר מדיי כדי להיות hash - לאחר בדיקה קצרה מסתבר שהוא חלק מהאש MD5 על השם של הקובץ. (1bb87d41d15fe27b500a4bfcde01bb0e) = logo.png

על מנת לאתר זאת, השתמשנו בכלי מהאתר כגון defuse.ca:

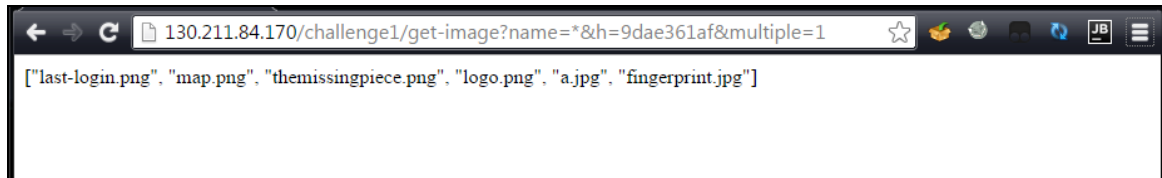
md5	1bb87d41d15fe27b500a4bfcde01bb0e
LM	9af381b3447232aedf128b2dd32bad07
NTLM	05dbddd6d4d378cf0d719a6fc73beda2
sha1	81c27284f77a447375ba39fb2f0005eeaccf28d8
sha256	ab211233b6576dbb0f8b5826447eeac61e2a833a99ac5d788fbc1a174c3c6ce5
sha384	c4685b7ac9058080939dd8baccf58ad4c10a4a7980e4155941ca8652979543f6f8847f24672009566e11fd5626f669e1
sha512	9fbe5c788fb7873905ce5898b708cc86a654489b31b8181a5de33133a94188b23cddcfa5189ed2dfa05c2ceb9ff041f99
md5(md5())	6b4542eb5a945f1a4716da6be501c28f
MySQL4.1+	3a064282c63867b9e6dc4701bce50db4fcc39403
ripemd160	e86bd172bb008154a7d8add8bb286c19be9d460a
whirlpool	0a662eadce50b333800ef66584bbd6ca7ed30752482d2d1a06edc1ceb98fd4ccac3126c5ace09c70857f6f6efc2ce38c0
adler32	0e510325
crc32	1f112a77

[מקור: <https://defuse.ca/checksums.htm>]



שימוש נכון בפרמטרים:

אם יש אפשרות לריבוי תכנים ויש לנו שם של קובץ נשמע שמדובר ב-Wildcard לכן אם נחליף את logo.png ל-* נשים את החלק הנכון של ההאש שלו ב-h ונכניס 1 ב-multiple אנחנו אמורים לקבל את מה שאנחנו מחפשים...



ואכן כך קרה ☺ קיבלנו את רשימת הקבצים שנמצאים באותה התיקיה. נכתוב סקריפט שיריד את כל הקבצים למחשב כדי שנוכל להבין איזה מהם יכולים לשמש אותנו ונדע גם את ה-CRC של כל אחד מהם:

```
import crc16
import json
import hashlib
import urllib2

def get_md5_hash(s):
    md5 = hashlib.md5()
    md5.update(s)
    return md5.hexdigest()

def get_default_index_size():
    to_find = '87d41d15f'
    md5_hash = get_md5_hash('logo.png')
    index = md5_hash.index(to_find)
    return index, index + len(to_find)

def get_file_url(name, show_name=1):
    start, end = get_default_index_size()
    md5_hash = get_md5_hash(name)
    return "http://130.211.84.170/challenge1/get-image?name=%s&h=%s&multiple=%d" % (
        name, md5_hash[start:end], show_name
    )

def get_list_of_all_files():
    res = urllib2.urlopen(get_file_url('*'))
    res_text = res.read()
    return json.loads(res_text)

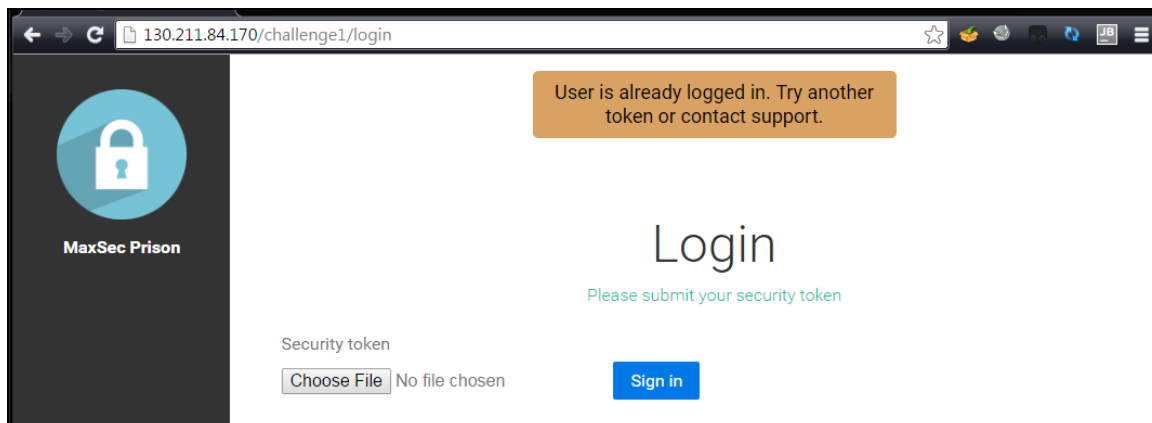
def get_file(name):
    file_url = get_file_url(name, show_name=0)
    res = urllib2.urlopen(file_url)
    return res.read()

if __name__ == '__main__':
    all_files = get_list_of_all_files()
    for f in all_files:
        f_data = get_file(f)
        print f, hex(crc16.crc16xmodem(f_data))
        with open(f, 'wb') as fb:
            fb.write(f_data)
            fb.flush()
```



קבצים מעניינים:

קובץ שמשך את תשומת ליבנו היה last-login.png. העלנו אותו ואכן התקדמנו - קיבלנו הודעת שגיאה חדשה:



כעת אנו יודעים שה-CRC שלו בסדר, מה שנשאר לעשות הוא לנסות להשתמש בקובץ אחר עם אותו ה-CRC!

על מנת לזייף CRC16 (הווה אומר: למצוא [התנגשות / Collision](#) או לייטר דיוק: [Preimage](#)) בצורה הפשוטה ביותר יש לעשות Bruteforce על 2 בייתים בסוף הקובץ: לכן נוסף 2 בייתים בכל הקומבינציות בסוף הקובץ עד שהקובץ יהיה עם אותו ה-CRC של הקובץ last-login.png:

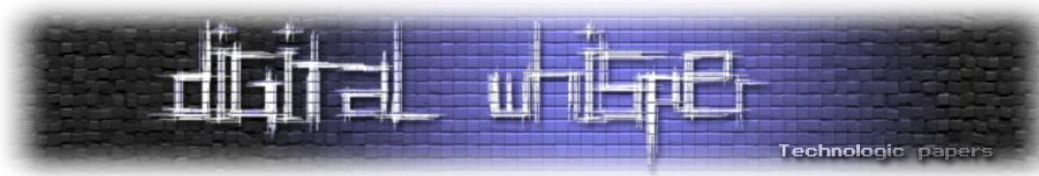
```
import crc16
from os import path
from itertools import combinations_with_replacement

ASCII_CHARS = map(chr, xrange(256))

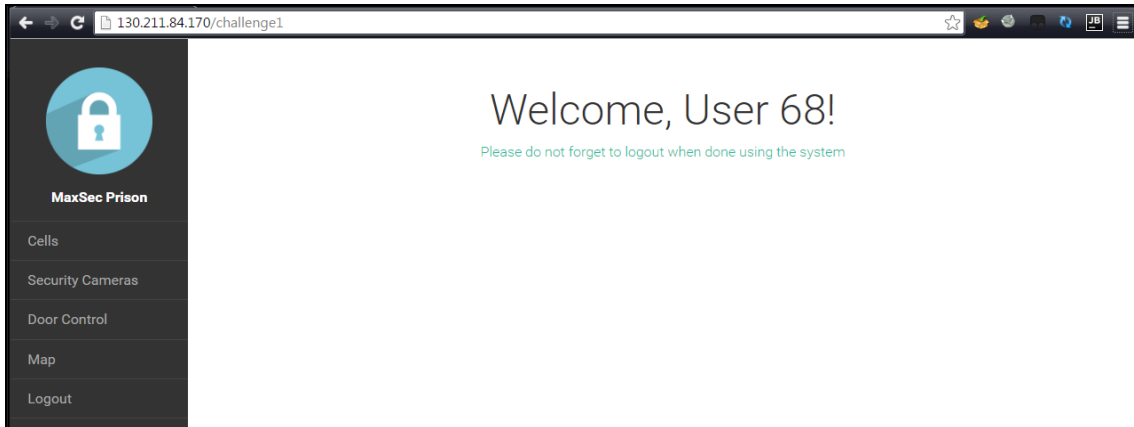
xmodem_target = int("398e", 16)
img_name = 'Realgame.png'
img_text = open(img_name, 'rb').read()

for i, suffix in enumerate(combinations_with_replacement(ASCII_CHARS, 2)):
    new_img_text = img_text + "".join(suffix)
    if xmodem_target == crc16.crc16xmodem(new_img_text):
        print 'xModem changed to 0x%04x after %d tries.' % (xmodem_target, i)
        with open("%s_xmodem%s" % path.splitext(img_name), 'wb') as img_file:
            img_file.write(new_img_text)
            img_file.flush()
            break
```

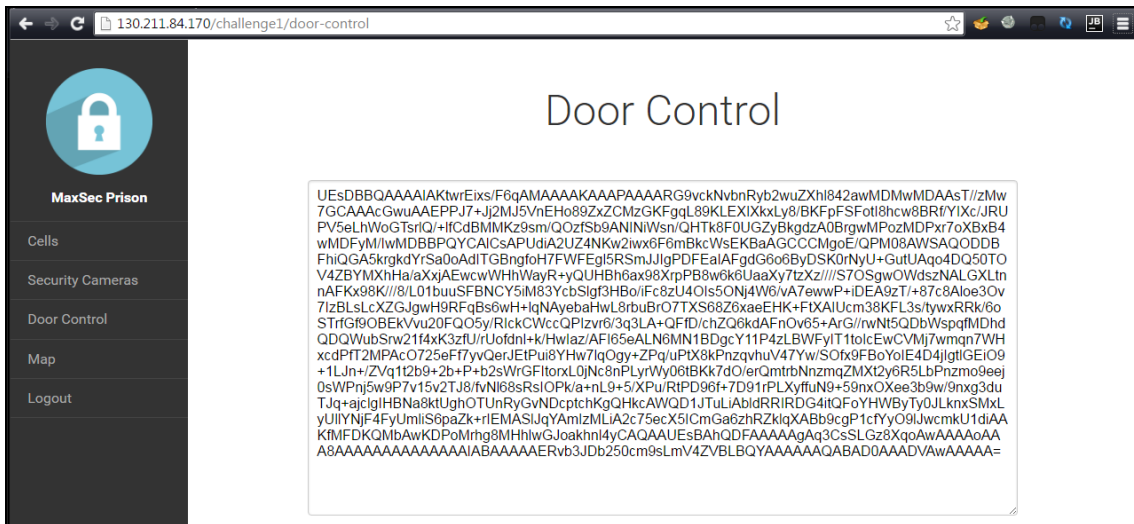
לאחר שהצלחנו לייצר תמונה עם ה-CRC הנכון, כל מה שנשאר זה - להעלות אותה.



העלנו את הקובץ לשרת ו... אכן הצלחנו להתחבר למערכת! הצלחנו להתחבר תחת משתמש מספר 68:



אוקיי... מה עכשיו? חיפוש קצר בתפריט יביא אותנו ל-Door Control:



כפי שניתן לראות יש תוכן שמקודד ב-BASE64. יצרנו סקריפט קצר שמקבל את התוכן ושומר אותו באופן בינארי באופן הבא:

```
import subprocess

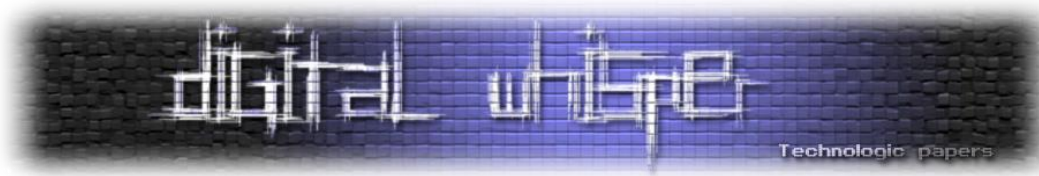
file_name = 'file.raw'

b64 = 'UESDBBQAAAAIAKtwrEixs/F6qAMAAAKAAAPAAARG9vckNvbnRyb2wuZXhi842awMDMwMDAAsT/zMw
7GCAAAcGwuAAEPPJ7+J2Mj5VnEHo89ZxZCMzGKfGqL89KLEIXkxLy8/BKfPFSFoti8hcw8BRfYIXc/JRU
PV5eLhWoGTrIQ/+fCdBMMKz9sm/QozfSb9ANINiWsn/QHTk8F0UGZyBkgdzA0BrgrwMPozMDPxr7oXBxB4
wMDFyM/iwMDBBPQYCAICsAPUdiA2UZ4Nkw2iwx6F6mBkcWsEKBaAGCCCMgoE/QPM08AWSAQODDB
FhiQGA5krkdYrSa0oAdITGBngfoH7FWFEgl5RSmJlJgPDFEalAFgdG6o6ByDSK0rNyU+GutUAqo4DQ50TO
V4ZBYMXhHa/aXjAEwcvWHhWayR+yQUHBh6ax98XrpPB8w6k6UaaXy7tzXz///S7OSgwOWdszNALGXLtn
nAFKx98K//8/L01buuSFBNCY5iM83YcbSlgf3HBo//Fc8zU4Ois5ONj4W6/vA7ewwP+iDEA9zT/+87c8Aloe3Ov
7lZBLsCXZGJgWH9RFqBs6wH+lqNAyebaHwL8rbuBrO7TXS68Z6xaeEHK+FXAIUcm38KFL3s/tywxRRk/6o
STrfG9OBekVvu20FQO5y/RickCWccQPizvr6/3q3LA+QFfD/chZQ6kdAFnOv65+ArG//rwNt5QDbWspqfMDhd
QQDQWubSrw21f4xK3zfU/rUofdnl+k/Hwlaz/AFi65eALN6MN1BDgcY11P4zLBWFyIT1tolcEwCVMj7wmqn7WH
xcdPFTZMPAcO725eF7yvQerJEIpu8YHw7lqOgy+ZPq/UPX8kPnzqvhUv47Yw/SOfx9FBoYolE4D4jgtIGEIO9
+1LJn+/ZVq12b9+2b+P+b2sWrGFItorxL0jNc8nPlyrWY06tBk7dO/erQmtrbNnmqZMXt2y6R5LbPnzmo9eej
0sWPnj5w9P7v15v2TJ8/fvNl68sRslOPka+nL9+5/XPu/RfPD96f+7D91rPLXyfuN9+59nxOXee3b9w/9nxg3du
TJq+ajciglHBNa8ktUghOTUnRyGvNDcptchKgQHkcAWQD1JTUliAbidRRIRDG4itQFoYHWBYTy0JLknxSMxL
yUIYNjF4FyUmliS6paZk+rEMASiJqAmzMLiA2c75ecX51CmGa6zhRZklqXABb9cgP1cYyO9JwcmkU1diAA
KfMFDKQMbAwKDPoMrhg8MHlwGJoaknl4yCAQAAUesBAHQDFAAAAAq3CsSLGz8XqoAwAAAAoAA
A8AAAAAAAAAAAAAAAAIABAAAAERbv3JDb250cm9sLmV4ZVBLBQYAAAAAQAAD0AAADAwAAAAA='
```

בדיקה קלה של הקובץ באמצעות הפקודה file בלינוקס תגלה לנו כי מדובר בקובץ Zip:

file.raw: Zip archive data, at least v2.0 to extract

פתרון אתגר המוסד 2016
www.DigitalWhisper.co.il



שמרנו את הקובץ עם סיומת zip ופתחנו – קיבלנו קובץ exe בשם: DoorControl.exe.

ברסינג ל-Door Control:

פתיחת הקובץ באמצעות IDA תלמד אותנו כי בקובץ יש 2 פונקציות. קטע הקוד הנ"ל מכיל את מספר הדלת שאנו צריכים לפתוח בכדי לעבור לשלב 2. בתמונה זו אנו רואים 2 מחרוזות בזיכרון שנשתמש בהן בהמשך (filename, keyInfo):

```
mov     esi, offset filename ; "c:\\doors\\config.txt"
lea     edi, [ebp+filename]
rep     movsd
mov     esi, offset keyInfo ; "c:\\OpenDoors.key"
lea     edi, [ebp+keyInfo]
push   25h
pop     ecx
xor     ebx, ebx
lea     eax, [ebp+var_97]
movsd  [ebp+keyInfoFromFile], b1
movsd  [ebp+keyInfoFromFile], b1
movsd  [ebp+keyInfoFromFile], b1
movsb  [ebp+keyInfoFromFile], b1
mov     esi, offset successUrl ; "http://130.211.84.170/challenge1/success"
lea     edi, [ebp+urlXorKey]
rep     movsd
movsw  [ebp+urlXorKey], w1
movsb  [ebp+urlXorKey], b1
call   sub_401168
```

המחרוזות בקוד מוצפנות עם XOR של בית אחד. הגודל של הקובץ הוא 19 (0x13) והבית שאיתו עושים XOR הוא 0xD8:

```
loc_401055: ; CODE XREF: start+5E1j
xor     byte ptr [ebp+eax+filename], 0D8h
inc     eax
cmp     eax, 13h
jl     short loc_401055
```

לאחר מכן אנו מנסים לקרוא את הקובץ c:\doors\config.txt (filename) עם שימוש בפונקציה CreateFileA:

```
text:00401060          push     ebx                ; hTemplateFile
text:00401060          push     ebx                ; dwFlagsAndAttributes
text:00401061          push     ebx                ; dwCreationDisposition
text:00401062          push     3                  ; lpSecurityAttributes
text:00401064          push     ebx                ; dwShareMode
text:00401065          xor     esi, esi
text:00401066          lea     ecx, [ebp+filename]
text:00401068          inc     esi
text:0040106B          push     esi                ; dwDesiredAccess
text:0040106C          push     ecx                ; lpFileName
text:0040106D          call   ds:CreateFileA
text:00401074
```

בקטע קוד זה אנו קוראים את הקובץ ובודקים קודם אם נקראו 16 בתים בקובץ, לאחר מכן משווים את התוכן שנקרא מהקובץ (c:\doors\config.txt) עם המחרוזת "(keyInfo) "c:\OpenDoors.key"

```

text:0040107D
text:0040107D      push     ebx                ; lpOverlapped
text:0040107E      lea     ecx, [ebp+lpNumberOfBytesRead]
text:00401081      push     ecx                ; lpNumberOfBytesRead
text:00401082      push     11h               ; nNumberOfBytesToRead
text:00401084      lea     ecx, [ebp+keyInfoFromFile]
text:0040108A      push     ecx                ; lpBuffer
text:0040108B      push     eax                ; hFile
text:0040108C      call    ds:ReadFile
text:00401092
text:00401092      test    eax, eax
text:00401094      jz      loc_40115F
text:0040109A
text:0040109A      cmp     [ebp+lpNumberOfBytesRead], 10h
text:0040109E      jnz     loc_40115F
text:004010A4
text:004010A4      mov     ecx, ebx
text:004010A6      loc_4010A6:                ; CODE XREF: start+BB↓j
text:004010A6      mov     al, [ebp+ecx+keyInfo]
text:004010AA      cmp     al, [ebp+ecx+keyInfoFromFile]
text:004010B1      jnz     loc_40115F

```

במידה והכל טוב אנו ממשיכים לקטע קוד הבא שמפענח את הכתובת URL לשלב 2, המפתח הוא XOR עם 0xAA:

```

text:004010BF loc_4010BF:                ; CODE XREF: start+D0↓j
text:004010BF      xor     [ebp+esi+urlXorKey], 0AAh
text:004010C7      jz      short loc_4010D2
text:004010C9
text:004010C9      inc     esi
text:004010CA      cmp     esi, 97h
text:004010D0      jb      short loc_4010BF

```

והכתובת URL היא:

http://130.211.84.170/challenge1/success/0496e88u_a9s6S7srqkvLiku76y9ru-vLizs7K-v7Okvr0=

הכתובת משתנה עבור כל מחשב לפי הכתובת IP ככה שה-Tokens יהיו שונים לכל בן אדם. אחרי שפענחנו את הכתובת URL ניתן להמשיך באמת לשלב הבא, אך עדיין אנו לא יודעים את מספר הדלת ואפשר להמשיך עוד קצת (בעת כתיבת המאמר, לאחר הפתרון, הסתבר כי בעמוד cells היו רשימה של מספרי דלתות (280-285) והיה ניתן לדעת את המטרה כבר בשלב זה).

אז לאחר מיקום כלל הקבצים שהבינארי צריך, נריץ אותו. כעת אנו מתבקשים להכניס מספר דלת כפי שמוצג בתמונה:

```

C:\Users\b4df00d\Desktop\reversing tools\mossad challalnge>DoorControl.exe
Enter cell number: _

```

אנו אמורים לקבל קלט מסוים בכדי לקבל את הכתובת במסך, הקטע קוד הבא אחראי לבדוק את מספר הדלת שאנו צריכים:

```

.text:00401117 ; // compare for door 281
.text:00401117      cmp     [ebp+door], '2'
.text:00401118      jnz     short loc_40110C
.text:0040111D      cmp     [ebp+door+1], '8'
.text:00401121      jnz     short loc_40110C
.text:00401123      mov     al, [ebp+door+2]
.text:00401126      sub     al, 30h
.text:00401128      cmp     al, 9
.text:0040112A      ja     short loc_40110C
.text:0040112C      push   0 ; lpReserved
.text:0040112E      lea   eax, [ebp+lpNumberOfCharsWritten]
.text:00401131      push  eax ; lpNumberOfCharsWritten
.text:00401132      push  0Dh ; nNumberOfCharsToWrite
.text:00401134      push  offset aDoorOpened ; "Door Opened\n\n"
.text:00401139      push  0FFFFFF5h ; nStdHandle
.text:0040113B      call  edi ; GetStdHandle
.text:0040113D      push  eax ; hConsoleOutput
.text:0040113E      call  ebx ; WriteConsoleA
.text:00401140      cmp     [ebp+door+2], '1'
.text:00401140      jnz     short loc_40115B

```

ברגע שאנו מכניסים את המספר דלת שיוצא 281, אנו נקבל את הפלט הבא:

```

C:\Users\b4df00d\Desktop\reversing tools\mossad challalnge>DoorControl.exe
Enter cell number: 281
Door Opened
http://130.211.84.170/challenge1/success/0496e88u_a9s6S7srqkvLiku76y9ru-vLizs7K-
v70kvr0=

```

אם נגלוש לעמוד - אכן נקבל את עמוד סיום השלב:

Success!

You have successfully completed Challenge #1.

This is your success token:

cb1ccecua_yvqS7urOku7K-pLy_9ru-vLm6vLuzuLmkvr8=

You may now [send](#) your token and CV

You may obtain additional tokens that will prove your skills by completing more challenges.

Take the Next Challenge



שלב שני - Knock Knock, Who's There ?

הסבר על המשימה:

Challenge #2

Well done!
Your colleague has been found and brought back home unharmed.
After initial debriefing, your colleague recalled a conversation about a communication system that enables the hostile group to communicate securely.
He remembered they said that the system is so sophisticated that it changes the auth details for each IP accessing it.
Your mission, should your skills suffice, is to gain access to the system.
Oh, and Agent C. next time you go on holiday, do let us know where you are heading...

Good luck,
A.

[Start Challenge](#)

לחיצה על הכניסה לשלב 2 מעבירה אותנו ישירות לעמוד הבא:

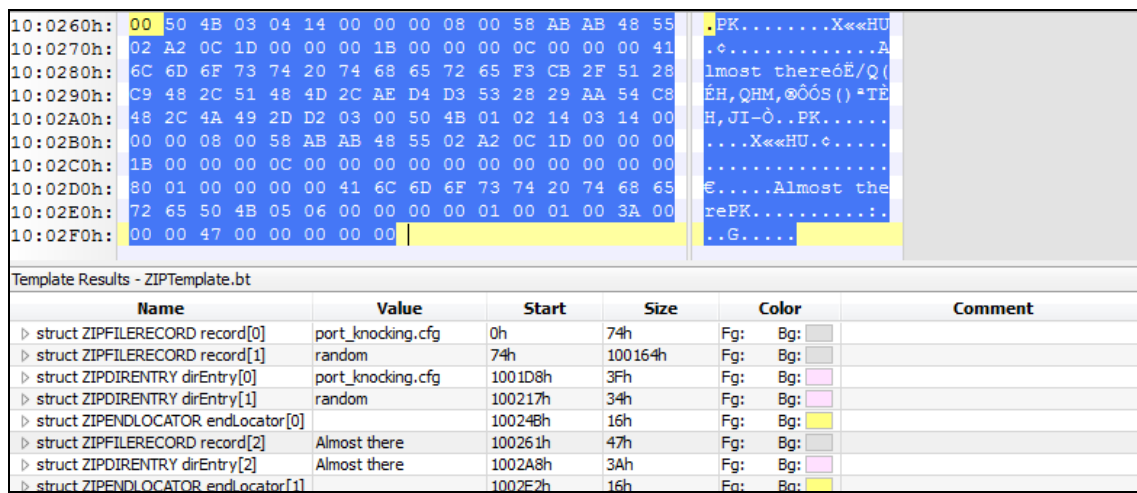


אחרי שדילגנו לקוד המקור של ה-challenge2/ (הדף שעשה Redirect לדף access-denied) ראינו שיש שם URI בהערת HTML. ומה שגרם ל-Redirect היה סקריפט:

```
<html lang="en">
<head>
<meta charset="utf8">
<link href='https://fonts.googleapis.com/css?family=Roboto:400,700,300' rel='stylesheet' type='text/css'>
<title>Challenge 2</title>
<link rel="stylesheet" href="/css/pure-min.css">
<link rel="stylesheet" href="/css/main.css">
</head>
<body>
<script>setTimeout(function() {
  window.location = '/challenge2/access-denied'
}, 0);</script>
<!-- /challenge2/pk -->
<script>(function(i,s,o,g,r,a,m){i['GoogleAnalyticsObject']=r;i[r]=i[r]||function(){
  (i[r].q=i[r].q||[]).push(arguments)},i[r].l=1*new Date();a=s.createElement(o),
  m=s.getElementsByTagName(o)[0];a.async=1;a.src=g;m.parentNode.insertBefore(a,m)
})(window,document,'script','https://www.google-analytics.com/analytics.js','ga');
ga('create', 'UA-77432143-1', 'auto');
ga('send', 'pageview');</script>
</body>
</html>
```

גלישה לקישור המוחבא מובילה אותנו לקובץ Zip בשם האינדיקטיבי: x.zip ששוקל כ-1mb. כאשר מחלצים את הקובץ x.zip אנחנו מקבלים קובץ טקסט קטנטן בשם Almost there. מיד שמנו לב שמשהו פה מוזר: הגודל של הקובץ הנ"ל קטן מאוד אך גודל הקובץ x.zip גדול מאוד יחסית לתוצאה.

פתיחת הקובץ עם Hex-Editor, מלמדת אותנו שלא מדובר בקובץ Zip אחד אלא 2 משורשרים! כותבי החידה ניצלו את העניין שמנגנון החילוץ של מערכת ההפעלה קורא את ה-Header הסופי של קבצי ה-Zip ועל-פיו מבין אילו קבצים יש לחלץ. במידה ונמחק את כל החלק של קובץ ה-Zip השני שמכיל את הקובץ Almost there באופן הבא:



Name	Value	Start	Size	Color	Comment
struct ZIPFILERECORD record[0]	port_knocking.cfg	0h	74h	Fg: Bg:	
struct ZIPFILERECORD record[1]	random	74h	100164h	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[0]	port_knocking.cfg	1001D8h	3Fh	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[1]	random	100217h	34h	Fg: Bg:	
struct ZIPENDLOCATOR endLocator[0]		10024Bh	16h	Fg: Bg:	
struct ZIPFILERECORD record[2]	Almost there	100261h	47h	Fg: Bg:	
struct ZIPDIRENTRY dirEntry[2]	Almost there	1002A8h	3Ah	Fg: Bg:	
struct ZIPENDLOCATOR endLocator[1]		1002E2h	16h	Fg: Bg:	

נקבל את התוצאה הרצויה הנ"ל:

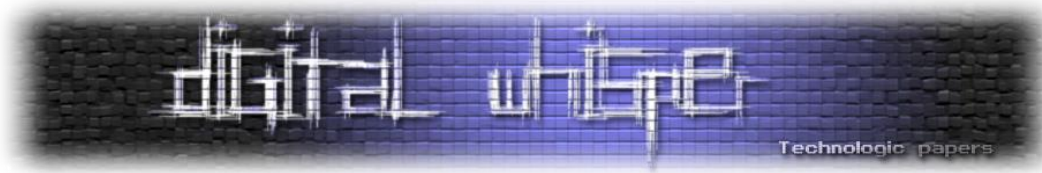
Name	Size	Packed Size	Modified	Created	Accessed	Attributes
port_knocking.cfg	75	69	2016-05-13 07:51			0rw-----
random	1 048 576	1 048 896	2016-05-13 07:51			0rw-----

כאן כבר הכל מובן, קיים קובץ נוסף המהווה את המשך החידה בשם: "port_knocking.cfg" וקובץ נוסף - שנועד לעזור לנו להבין שיש כאן קאצ', בשם: "random" ששוקל כ-1mb... נמשיך. אז מצאנו קובץ מעניין בשם: port_knocking.cfg (נראה כמו קובץ קונפיגורציה ל-Port Knocking):

```
port_knocking.cfg x
1 [Port Knocking]
2 timeout: 1000ms
3 knock_ports: 6466,6404,6574
4 dest_port: 1337
```

למי שמעוניין להרחיב בעניין ה-Port Knocking - מוזמן לקרוא את המאמר הנ"ל:

<http://www.digitalwhisper.co.il/files/Zines/0x02/DW2-5-Port-Knocking.pdf>



נכתוב ב-Python כלי זריז שישתמש בקונפיגורציה הנ"ל:

```
import socket
from functools import partial
from configparser import ConfigParser

def recvall(sock):
    data = []
    while True:
        try:
            data.append(sock.recv(1024))
        except socket.timeout:
            break
    return b"".join(data)

def knock(ip, port):
    print ip, port
    try:
        s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
        s.connect((ip, port))
        s.close()
    except socket.timeout:
        pass

if __name__ == '__main__':
    config = ConfigParser()
    with open('port_knocking.cfg') as config_file:
        config.readfp(config_file)
        options = dict(config.items('Port Knocking'))

    # Parse The Config
    knock_ports = map(int, options['knock ports'].split(','))
    timeout = int(options['timeout'].rstrip('ms')) / 1000.0
    dest_port = int(options['dest_port'])

    # Set timeout
    socket.setdefaulttimeout(timeout)

    # Knock ports
    map(partial(knock, '130.211.84.170'), knock_ports)

    # Connect to the destination
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('130.211.84.170', dest_port))

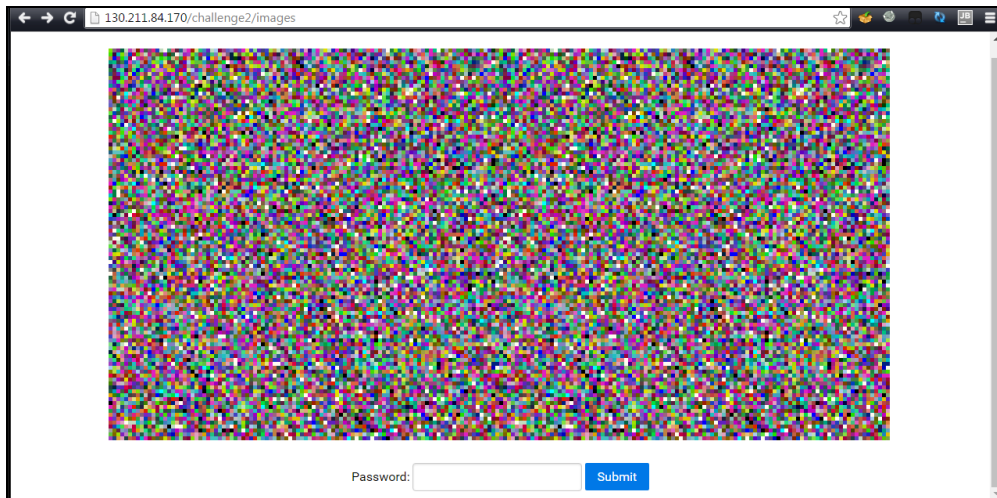
    # Print welcome message
    welcome = recvall(s)
    print welcome,

    # Shell
    while True:
        msg = raw input()
        s.sendall(msg + b"\r\n")
        print recvall(s),
```

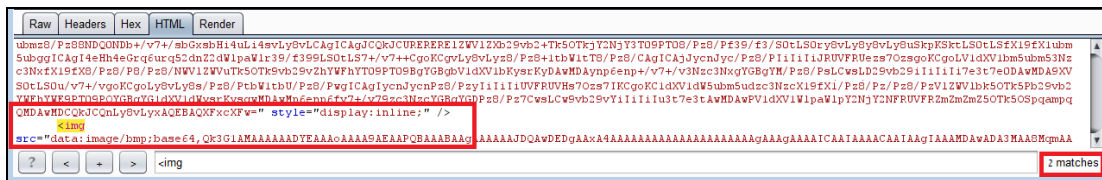
ואכן - הכל עבד חלק ☺

התחברנו לשרת בהצלחה ועכשיו יש לנו Shell מרוחק אשר מאפשר לנו לכתוב פקודות, פקודה שתפסה את העין במיוחד הייתה הפקודה hdump. מבדיקה על מספר קבצים בשרת נראה שהיא מציגה את תוכן הקבצים כ-hex.

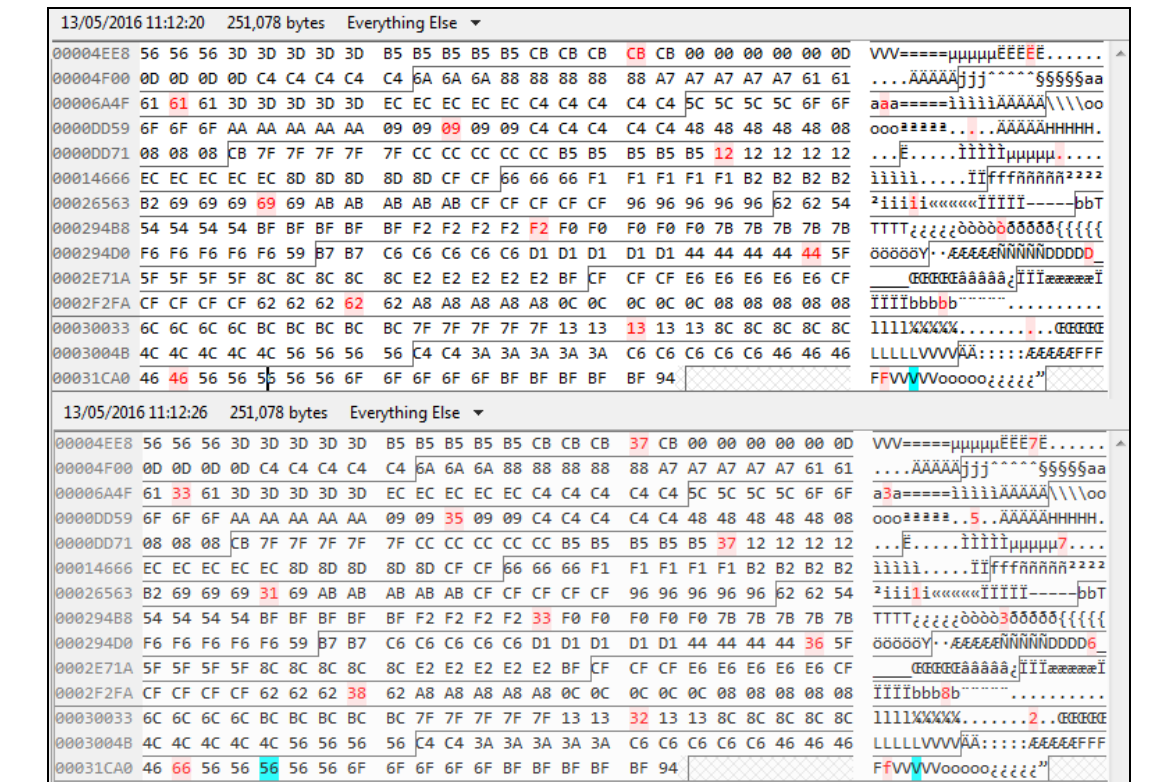
גלישה לעמוד הנ"ל מביאה אותנו לעמוד הבא:



על פניו - נראה כי מדובר בעמוד עם תמונה מוזרה ותיבת טקסט המצפה לקבל סיסמה. אם נביט בקוד המקור נגלה שזו לא תמונה אחת אלא 2 תמונות שונות אחת ליד השנייה (display:inline):



נוריד את 2 התמונות למחשב ונשווה ביניהן:





אפשר לראות כי בצד שמאל יש לנו בהבדלים תווי ASCII שהם קריאים, נכתוב קוד קצר ב-Python שידיפס לנו את ההבדל הקריא למסך:

```
img_right = open('right.bmp', 'rb').read()
img_right_diff = []
img_left = open('left.bmp', 'rb').read()
img_left_diff = []

for i in xrange(len(img_left)):
    if img_left[i] != img_right[i]:
        img_right_diff.append(img_right[i])
        img_left_diff.append(img_left[i])

img_right_str = "".join(img_right_diff)
img_left_str = "".join(img_left_diff)

if img_left_str.isalnum():
    print img_left_str
else:
    print img_right_str
```

נחבר את התווים האלה ביחד ויצא לנו: 735713682f. הכנסנו את המחרוזת הנ"ל לתיבת הטקסט - וזו אכן הייתה הסיסמה!

Success!

You have successfully completed Challenge #2.
This is your success token:
524a378uPayvqS7urOku7K-pLy_9ru-vLm6sr08s72ksr8=

You may now [send](#) your token and CV
You may obtain one more token by completing the final challenge.

Take the Final Challenge

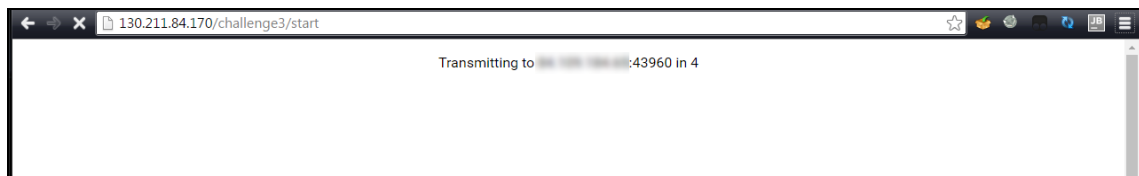
Challenge #3

Amazing job!
Now we have access to the hostile group's communication system and we can intercept their messages. Unfortunately our cipher specialist is on maternity leave, and we need you for one last task.
Your mission is to decipher the message we have intercepted and give us critical information as to the group's next move.

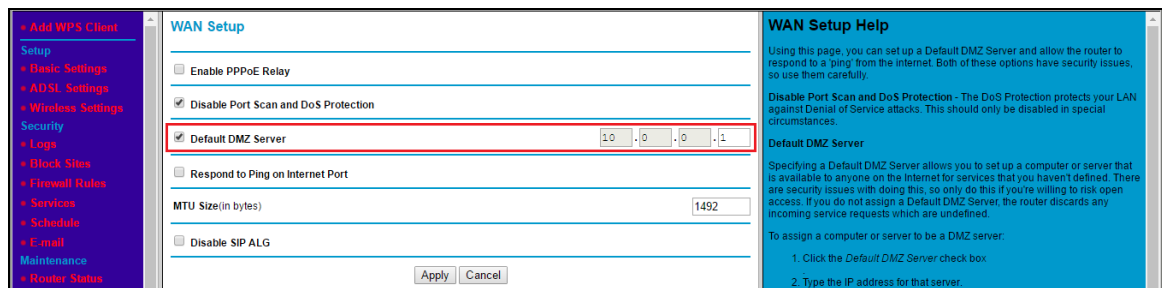
Good luck, we are counting on you,
A.

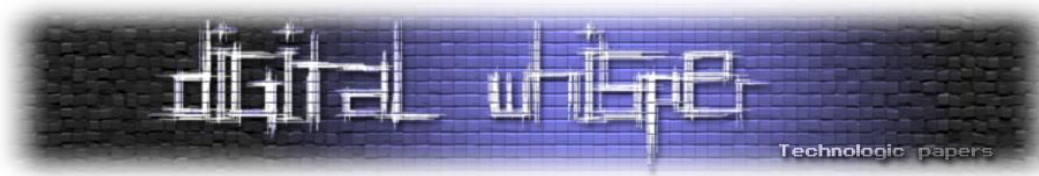
Start Challenge

בתחילת השלב נראה כי שרת האתגר מעוניין ליצור עם כתובת ה-IP שלנו חיבור בפורט משתנה:



על מנת לקבל את החיבור ולראות את תוכן המידע שישלח אלינו נפתח את עצמנו זמנית ל-DMZ בראוטר הביתי (הפתרון הנ"ל עדיף על השימוש ב-Port Forwarding מפני שהוא מאפשר לנו לקבל כל פורט שנרצה ולא רק פורט או טווח ספציפי), כך זה נראה ב-NetGear:





לאחר שעשינו זאת, עלינו להאזין לפורט ששרת האתגר יבחר. נפתח את Netcat בעזרת הפקודה:

```
nc -l -p 43960 > out.file
```

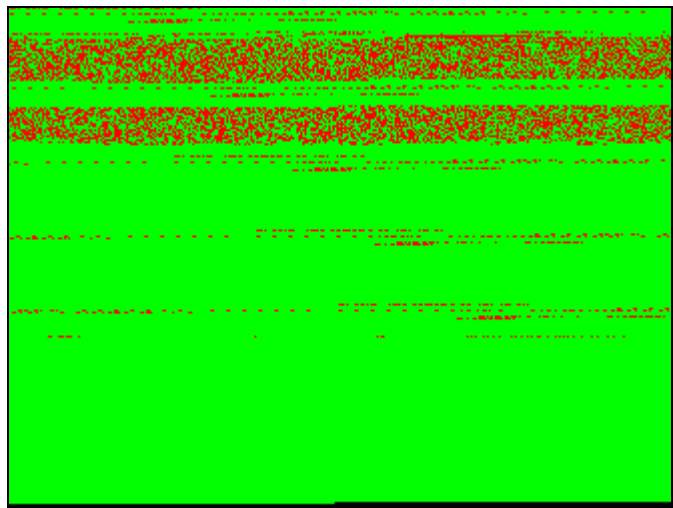
- -l מצב האזנה.
- -p פורט להאזנה.
- -> הפניית הפלט לתוך קובץ.

לאחר מספר נסיונות הצלחנו להאזין על הפורט הנכון (יש 5 שניות להרים את ה-netcat הנ"ל). כעת נסתכל מה יש בקובץ שנוצר!

נבדוק מה סוג הקובץ שקיבלנו בעזרת file ונגלה שמדובר בקובץ תמונה (png):

```
root@kali:~/Desktop# file out.file
out.file: PNG image data, 331 x 251, 8-bit/color RGB, non-interlaced
```

נשנה את סיומ הקובץ ל-png ונפתח אותה, זו התמונה שקיבלנו (משתנה בין מחשב למחשב):



התמונה דומה קצת לסוג של ברקוד בצבעים ירוק ואדום... אולי אדום זה 1 וירוק זה 0 ומדובר בקוד בינארי? נכתוב קוד ב-Python שיאשר את זה:

```
from PIL import Image

def chunks(l, n):
    """Yield successive n-sized chunks from l."""
    for i in range(0, len(l), n):
        yield l[i:i + n]

bit_map = {
    (255, 0, 0): '1',
    (0, 255, 0): '0',
}

im = Image.open("f.png")
width, height = im.size
```



```
def get_bit():
    for y in xrange(height):
        for x in xrange(width):
            pixel = im.getpixel((x, y))
            if pixel not in bit_map:
                continue
            yield bit_map[pixel]

byte_arr = []
for b in chunks(list(get_bit()), 8):
    byte_arr.append("%02x" % int("".join(b), 2))

with open('f', 'wb') as f:
    f.write("".join(byte_arr).decode('hex'))
    f.flush()
```

בדיקה נוספת באמצעות file על סוג הקובץ החדש מלמדת אותנו כי מדובר בקובץ tar. נחלץ אותו באמצעות הפקודה:

```
tar -xvf file
```

ונקבל:

```
root@kali:~/Desktop# file f
f: POSIX tar archive (GNU)
root@kali:~/Desktop# tar -xvf f
tmp/concat.1
tmp/concat.2
tmp/concat.3
tmp/concat.4
tmp/concat.5
```

כעת, אנחנו במצב שיש לנו תיקייה עם 5 קבצים שונים בשם concat, נזרום עם הרמז שקיבלנו ו... נחבר אותם ביחד בעזרת הפקודה cat:

```
cat concat.* > concat
```

יש לנו קובץ חדש בשם concat. נריץ על הקובץ החדש את הפקודה file, ונגלה שהוא מסוג: cpio archive. נחלץ גם אותו בעזרת הפקודה: cpio -idvF concat:

```
root@kali:~/Desktop/tmp# cpio -idvF concat
tmp/file
9 blocks
```

לאחר החילוץ קיבלנו קובץ נוסף. הפעם בשם file. הרצת הפקודה file עליו לימדה אותנו כי מדובר בקובץ מסוג: Squashfs filesystem.

למי שמעוניין להרחיב בקריאה על Squashfs יכול לקרוא בקישור הבא:

<https://en.wikipedia.org/wiki/SquashFS>



נשתמש ב-squashfs-tools כדי לחלץ את הקבצים מה-Image בעזרת הפקודה: unsquashfs file:

```
root@kali:~/Desktop/tmp/tmp# unsquashfs file
Parallel unsquashfs: Using 1 processor
1 inodes (1 blocks) to write

[=====|] 1/1 100%
created 1 files
created 1 directories
created 0 symlinks
created 0 devices
created 0 fifos
root@kali:~/Desktop/tmp/tmp# ls
file squashfs-root
```

ניכנס לתיקייה **squashfs-root** ונמצא שם קובץ ששמו **file.png**, פתיחת הקובץ מראה לנו את התמונה הבאה:

```
http://130.211.84.170/challenge3/success/c6c4c09469eec1a4e292be04b1c49d60
```

גלישה לעמוד הנ"ל ונראה שסיימנו לפתור את האתגר!

Success!

You have successfully completed the final challenge!
This is your success token:
a8d0115ufayvqS7ur0ku7K-pLy_9ru-vLm7ubq6vLikv7w=

You may now [send](#) your token and CV
Looking forward to see you in our next challenges.



סיכום ומספר מילים על האתגר

האתגר היה מגוון מאוד ודרש ידע בכמה וכמה תחומים. כגון: Reverse ,Web Application Security , Operation Systems,Engineering, תכנות רשתות ועוד. נראה כי הוא עבד כמו שצריך גם כשמספר המשתמשים שהשתתפו בו היה רב.

לדעתנו, השלב הראשון היה השלב המאתגר ביותר, הוא דרש יכולות Penetration עם הבנה של הרמזים הנלווים כמו ה-XMODEM-CRC והודעת השגיאה שקורת כשמעלים את הקובץ last-login.png, יכולת ברסינג בסיסית כדי להבין מה הדלת הנכונה ועוד.

השלב השני היה מעין חידה, הוא היה יותר פשוט כיוון שהחלק עם קובץ ה-Zip הפגום לא היה אצל כולם, דבר שהקל על המשימה (7-Zip החדש פתח את הקובץ בלי בעיה).

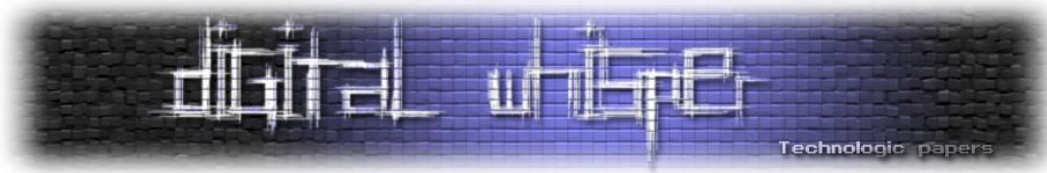
בשלב השלישי הגיוני שאנשים נתקעו לא מעט בגלל פתיחת הפורטים. עם ידע בינוני בתכנות היה אפשר להקל על התהליכים הידניים ובחלק מהמקרים ללא ידע בסיסי לא היה אפשר לעבור שלב (כגון במקרה של הפרסור תמונה).

אנו מקווים שנהנתם מקריאת המאמר לפחות כפי שאנו נהגנו לפתור את האתגר 😊 בתקווה שיהיו אתגרים נוספים כאלה בעתיד...

5ee y0u n3xt 7ime!

Thank you all for participating in the 2016 Independence Day challenge. We hope you enjoyed it. See you at the next challenge!

- The Mossad Operational Cyber Section



קישורים בנושא

- <https://en.wikipedia.org/wiki/XMODEM>
- https://en.wikipedia.org/wiki/Port_knocking
- <https://defuse.ca/checksums.htm>
- <https://pillow.readthedocs.io/en/3.2.x>
- <https://users.cs.jmu.edu/buchhofp/forensics/formats/pkzip.html>

על המחברים

- **D4D**: עוסק בתחום ה-Reverse Engineering - בחברת איירון סורס במחלקת ה-Security ואוהב לחקור משחקי מחשב והגנות, לכל שאלה שיש או ייעוץ ניתן לפנות אלי דרך:
 - שרת ה-IRC של Nix בערוץ: #reversing
 - או באתר: www.cheats4gamer.com
 - או בכתובת האימייל: llcashall@gmail.com.
- **תומר זית (RealGame)**: חוקר אבטחת מידע בחברת F5 Networks וכותב Open Source.
 - אתר אינטרנט: <http://www.RealGame.co.il>
 - אימייל: realgam3@gmail.com
 - GitHub: <https://github.com/realgam3>

משטחי תקיפה באפליקציות Android - חלק א'

מאת 0x3d5157636b525761

הקדמה

זהו המאמר הראשון של סדרת מאמרים שתציג משטחי תקיפה (Attack surfaces) לאנדרואיד. נתמקד בעיקר בקוד האפליקציות (זה שכתוב ב-Java) אבל יש סיכוי שנבצע "זליגה" מפעם לפעם לקוד native פגיע.

בהמשך המאמר נציג כיצד ניתן לבצע פעולות מרוחקות על אפליקציית WheresMyDroid הפופולרית (בין 10 ל-50 מיליון התקנות), כולל שדרוג שלה (שאמור לעלות כסף) וכן פעולות שונות על המכשיר.

השתלשלות האירועים

- 20.04.2016 - גילוי הנקודות (שתוצגנה בהמשך) ב-WheresMyDroid.
- 21.04.2016 - פנייה אל מפתחי האפליקציה. לצערי הם לא הגיבו.
- 01.05.2016 - פנייה נוספת אל מפתחי האפליקציה. עדיין לא הגיבו.
- 07.05.2016 - פרסום (Public disclosure).

רקע בסיסי על אנדרואיד

בשלב זה נספק רקע בסיסי (ביותר) על אנדרואיד. הלו המכירים את הארכיטקטורה מוזמנים לדלג הלאה.

אפליקציות ו-Dalvik

משתמשי אנדרואיד מפעילים אפליקציות. בניגוד לתהליך מסורתי על מערכת הפעלה, "אפליקציה" יכולה לרוץ מכמה תהליכים, ובדרך כלל יש לה יותר מ-Entry point יחיד. אפליקציות מגיעות ב-Archive עם סיומת APK (בפועל, ניתן לפתוח אותן כמו כל zip רגיל). בתוך ה-APK יש מספר פריטים מעניינים:

- **AndroidManifest.xml** - זהו קובץ המכיל את כל ה-Metadata של האפליקציה, ובפרט את ההרשאות שבו האפליקציה משתמשת (כגון כתיבה ל-SD או קריאה של SMS) וכן את ה-Entry points שלה.

משטחי תקיפה באפליקציות - Android חלק א'

www.DigitalWhisper.co.il



לאפליקציה יכולות להיות מספר Entry points מסוגים שונים:

- **Activity** - ה-Entry point הנפוץ ביותר, המציין אלמנט GUI שמשמש יכול לעבוד איתו (די דומה ל-form).
- **Service** - חלק באפליקציה שנועד לבצע פעולות ממושכות ברקע (למשל, לנגן מוזיקה).
- **Content Provider** - כל מה שעלול לספק תוכן לאפליקציות אחרות. אף על פי שאנדרואיד מגיעה עם מספר Content providers שלה, אין מניעה מלייצר כאלה.
- **Broadcast Receiver** - כל מה שיכול להתעדכן כתוצאה מ-Content provider אחר.
- **classes.dex** - כאן נמצאת (כמעט) כל הלוגיקה של האפליקציה. כל ה-class-ים נמצאים ממש כאן. קוד לאנדרואיד נכתב באופן טיפוסי בג'אווה (מעל מימוש בשם Apache Harmony), אך מקומפל ל-bytecode מיוחד בשם Dalvik (או "ART" על פלטפורמות חדשות). אנדרואיד מכילה JVM מיוחד בשם DVM שידוע להריץ את ה-bytecode הזה.
- **META-INF** - מכילה חתימות דיגיטליות (באופן זהה לקבצי JAR). אנדרואיד מחייבת כל APK להיות חתום עם חתימה כלשהי.
- **lib** - בתיקה זו ישבו קבצי SO למיניהם. אנדרואיד מאפשרת ממשק בין Java לקוד native (שנכתב בדרך כלל ב-C או C++). ממשק זה ידוע בתור JNI, ומשמש באפליקציות רבות (כגון Facebook, Chrome ו-WhatsApp). אף על פי שאנדרואיד רצה על מעבדי ARM, תחת lib תופענה תתי תיקיות המציינות את שם הארכיטקטורה (כגון x86, ARMv7 וכדומה). יש לציין שרוב האפליקציות לא מכילות סתם ככה קוד native - הסיבות לקוד כזה הוא בדרך כלל ניסיון להשיג ביצועים טובים יותר (כגון מימוש RTP של WhatsApp) או שימוש בחבילות סגורות (כגון BoringSSL עבור Chrome).

מערכת ההרשאות

כמו בכל מערכת לינוקס, לכל קובץ ישנן הרשאות. ההרשאות הללו מציינות אילו ישויות יכולות לבצע קריאה (Read), כתיבה (Write) או הרצה (execute) של הקובץ. הישויות הן בעל הקובץ (User), קבוצת הבעלות של הקובץ (Group) ואחרים (Others). את ההרשאות משנים על ידי הפקודה chmod (ישנן פקודות מתאימות גם לשינוי ה-owner של הקובץ, למשל).

מתחת לפני השטח, אנדרואיד עצמה מייצרת User עבור כל אפליקציה. בטרמינולוגיה של אנדרואיד, נוצר Application ID חדש ("AID"), אף על פי שמדובר במשתמשי לינוקס לכל דבר ועניין. רוב ההרשאות של האפליקציה (כפי שצוינו בקובץ ה-manifest שלה) תתורגמה לשייכות לקבוצות כאלה ואחרות (למשל, ישנה קבוצה שכל מי ששייך אליה יכול לכתוב אל כרטיס ה-SD). בנוסף, בוצעו שינויים מסויימים בקרנל כדי לתמוך בהרשאות מסויימות (למשל, אכיפה על כך שרק מי ששייך לקבוצה מסויימת יוכל לפתוח socket).

תקשורת בין תהליכים (IPC)

אנדרואיד "הפשיטה" את רוב מנגנוני ה-IPC המסורתיים של לינוקס ומימשה דרייבר בשם ה-Binder. דרייבר זה אחראי לחלק גדול של ה-IPC במערכת. הארכיטקטורה של ה-Binder די מרתקת (ומזכירה במקצת ממשקי COM), ולמזלנו רוב כותבי האפליקציות לא צריכים לדבר עם הדרייבר ישירות. ישנן אבסטרקציות רבות לעבודה מול ה-Binder, כאשר האבסטרקציה הנפוצה ביותר ידועה בתור Intent.

ניתן לשלוח Intent אל אפליקציה ספציפית, בין רכיבים שונים של אפליקציה, או פשוט "לכל מי שמוכן לקבל את ה-Intent". כמובן, Intent יכול להכיל בתוכו מידע, וכך למעשה שליחת Intent-ים מעל ה-Binder נותנת לכותב אפליקציה המון כוח לבצע IPC כמעט ללא טיפול בכאבי ראש בדמות סנכרון או סיראליזציה.

כאשר שולחים Intent אל אפליקציה ספציפית ה-Intent ידועה בתור Explicit intent, וכאשר שולחים "לכל מי שעונה על קריטריון כלשהו" אז ה-Intent ידועה בתור Implicit intent. באופן כללי, Implicit intents נפתרים על ידי הגדרת פילטרים (Intent filters), שבדרך כלל מוגדרים ב-AndroidManifest.xml.

מנגנונים נוספים

ישנם מנגנונים נוספים מעניינים במערכת שכדאי לציין כבר בשלב זה:

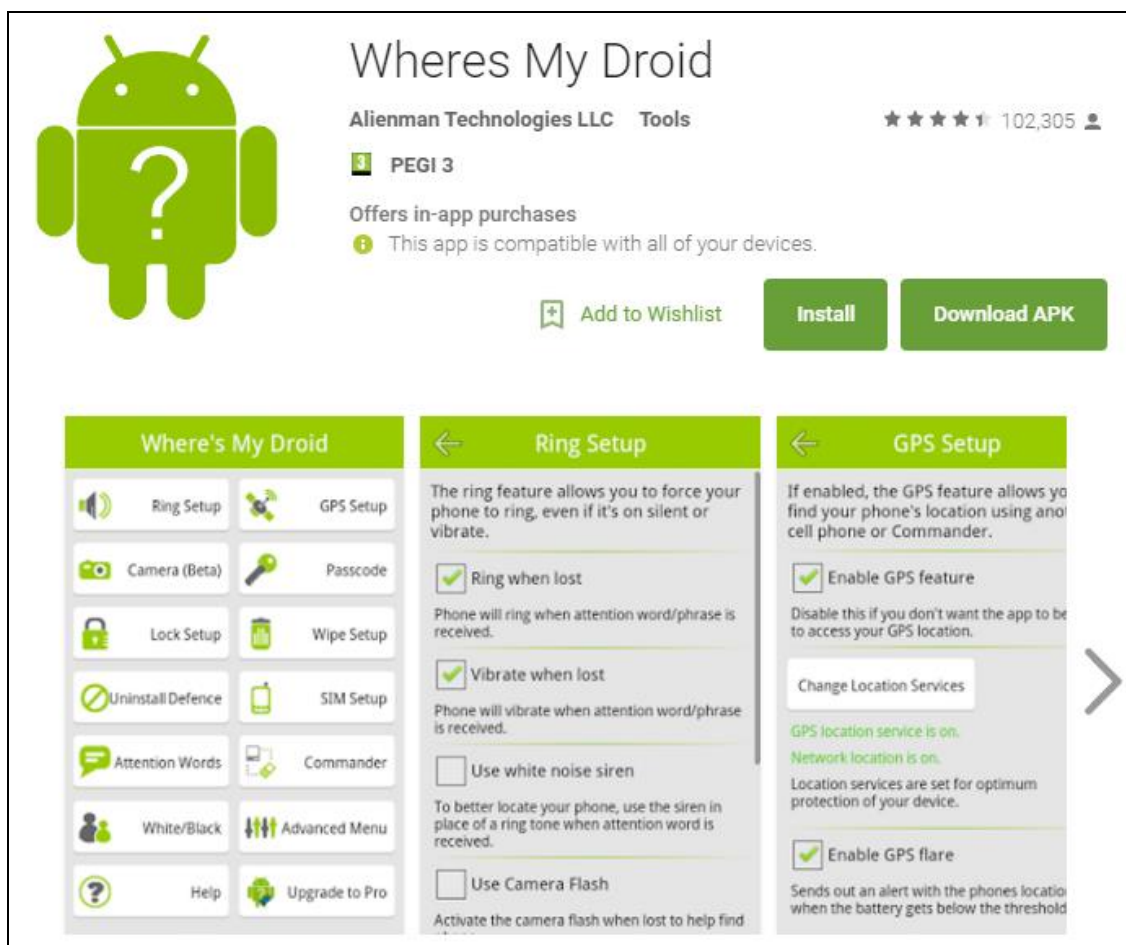
- לאנדרואיד יש גרסה משלה ל-SELinux (ידועה בשם SEAndroid). זה אומר שגם משתמש Root לא כל-יכול.
- באנדרואיד, הרשאות מסוימות יכולות להינתן רק לאפליקציות שמוקנות תחת ה-System partition (בניגוד לאפליקציות רגילות שמוקנות תחת ה-Data partition). האפליקציות הללו ידועות בתור System-apps, ועקרונית אין דרך להסיר אותן או להתקין חדשות כאלה אלא אם כן מבצעים Rooting למכשיר. ההרשאות המיוחדות הללו נותנות כוח רב לאפליקציות System - ביניהן היכולת להתקין APK חדש ללא התרעה ("התקנה שקטה") וכן קריאת לוגים של המערכת (logcat), שאליהם נכתב המון מידע שימושי ומעניין.

משטח תקיפה מבוסס SMS

משטח התקיפה הראשון שבו נתבונן הוא SMS. נדמה לעיתים כי מתכנתי אפליקציות לא חושבים על SMS כעל משטח תקיפה לגיטימי (בניגוד לגלישה לדפי אינטרנט, למשל), ולכן פעמים רבות סומכים על תוכן ה-SMS ללא עוררין.

לצורך ההדגמה, נבחן את אפליקציית WheresMyDroid, המותקנת על 10 מיליון - 50 מיליון מכשירים. מטרת האפליקציה היא לאתר את המכשיר במקרה והוא אבד או נגנב.

להלן פרטי האפליקציה, כפי שמופיעים ב-Google play:



ADDITIONAL INFORMATION		
Updated	Size	Installs
August 28, 2015	3.0M	10,000,000 - 50,000,000
Current Version	Requires Android	Content Rating
5.2.7	2.3 and up	PEGI 3 Learn more

הורדת ה-APK יכולה להתבצע במספר דרכים:

1. שליפת ה-APK מתוך ה-filesystem לאחר ההתקנה (דורש אסקלציה).
2. ביצוע Man in the middle על ה-Google services (דורש התקנה של סרטיפיקט ועדיין לא טרוויאלי).
3. הורדת ה-APK מאתר צד ג' בחינם.

משטחי תקיפה באפליקציות - Android חלק א'

www.DigitalWhisper.co.il



ישנם חוקרים שלא מביעים אמון באתרי צד ג' מסוג זה, אך מנסיוני האישי אני יכול להגיד בפה מלא שרובם מהימנים. עם זאת, הייתי ממליץ תמיד על סביבת עבודה נקייה (כלומר, לא לחקור אפליקציה על המכשיר האישי שלנו). אני משתמש באופן אישי ב-apk-pure.com, אבל ישנם שירותים דומים נוספים כמובן.

לאחר שהורדנו את ה-APK, ניתן להתחיל לחקור אותו באופן סטטי. ישנם כלים טובים לכך:

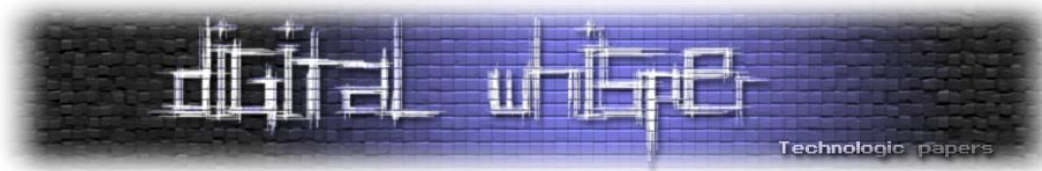
- **apktool** - כלי מצויין זה מתרגם את ה-AndroidManifest.xml ל-XML אמיתי (ה-manifest המקורי הוא בינארי על אף הסיומת שלו), שולף resources וכן מבצע תרגום של classes.dex ל-SMALI code. SMALI code הוא על פי שניצין מהו SMALI code במאמרים הבאים, ניתן להגיד כי היחס בין SMALI ל-Dalvik bytecode הוא כמו היחס בין אסמבלי לשפת מכונה. נוסף ונציין כי apktool יודע לבנות מחדש (repacking) קבצי APK.
- **dex2jar** - מתרגם APK שלם ל-JAR. כמובן, חלק מהמידע הולך לאיבוד (למשל, ה-manifest).
- **jd-gui** - לא באמת כלי עבור אנדרואיד, אבל כלי זה מהווה Java disassembler ויודע לעבוד עם קבצי JAR.
- **JEB** - כלי חקירה all-in-one עבור אנדרואיד. לא חינומי (בניגוד לכלים הקודמים שהזכרתי). אני התרגלתי להשתמש בכלים החינומיים, אבל JEB נחשב לאיכותי ביותר - שווה לבדוק אותו.

כעת ניתן לגשת לניתוח הראשוני. השלב הראשון יהיה להריץ apktool:

```
D:\research\WheresMyDroid>java -jar D:\1337\apktool\apktool_2.1.0.jar d -o apktoolout WheresMyDroid_v5.2.7.apk
I: Using Apktool 2.1.0 on WheresMyDroid_v5.2.7.apk
I: Loading resource table...
I: Decoding AndroidManifest.xml with resources...
I: Loading resource table from file: C:\Users\USER\apktool\framework\1.apk
I: Regular manifest package...
I: Decoding file-resources...
I: Decoding values **/*.XMLs...
I: Baksmaling classes.dex...
I: Copying assets and libs...
I: Copying unknown files...
I: Copying original files...

D:\research\WheresMyDroid>dir apktoolout
Volume in drive D has no label.
Volume Serial Number is 8A1F-2AE0

Directory of D:\research\WheresMyDroid\apktoolout
05/07/2016 11:17 AM <DIR>      -
05/07/2016 11:17 AM <DIR>      ..
05/07/2016 11:17 AM             12,049 AndroidManifest.xml
05/07/2016 11:17 AM             406 apktool.yml
05/07/2016 11:17 AM <DIR>      assets
05/07/2016 11:17 AM <DIR>      original
05/07/2016 11:17 AM <DIR>      res
05/07/2016 11:17 AM <DIR>      smali
                2 File(s)          12,455 bytes
                6 Dir(s)      179,415,449,600 bytes free
```



לאחר הרצה מוצלחת, נוכל להתבונן ב-AndroidManifest.xml:

```
<uses-permission android:name="com.android.vending.BILLING" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.BATTERY_STATS" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.FLASHLIGHT" />
<uses-permission android:name="android.permission.GET_ACCOUNTS" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.PROCESS_OUTGOING_CALLS" />
<uses-permission android:name="android.permission.READ_CALL_LOG" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.RECEIVE_SMS" />
<uses-permission android:name="android.permission.SEND_SMS" />
<uses-permission android:name="android.permission.USE_CREDENTIALS" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.WRITE_SETTINGS" />
```

המון הרשאות - נראה מבטיח! שימו לב במיוחד להרשאות SEND_SMS ו-RECEIVE_SMS שימשו אותנו. ניתן לראות גם כי קיים Broadcast receiver בשם SMSReceiver שייקרא בכל פעם שנקבל SMS:

```
<receiver android:name=".receivers.SMSReceiver">
  <intent-filter android:priority="999">
    <action android:name="android.provider.Telephony.SMS_RECEIVED" />
  </intent-filter>
</receiver>
```

כעת כדאי לבחון את הקוד של ה-Receiver הזה באמצעות dex2jar (ולאחר מן jd-gui):

```
D:\research\WheresMyDroid>D:\1337\dex2jar\dex2jar-0.0.9.15\d2j-dex2jar -o d2j.jar WheresMyDroid_v5.2.7.apk
dex2jar WheresMyDroid_v5.2.7.apk -> d2j.jar
```


הפונקציה onReceive תיקרא עם ה-Intent הרלוונטי. להלן הפלט של dex2jar (הקוד קטוע אך ממשיך עוד):

```
public void onReceive(Context paramContext, Intent paramIntent)
{
    String str1 = "";
    String str2 = "";
    boolean bool = false;
    try
    {
        Bundle localBundle1 = paramIntent.getExtras();
        Object[] arrayOfObject;
        SmsMessage[] arrayOfSmsMessage;
        if (localBundle1 != null)
        {
            arrayOfObject = (Object[])localBundle1.get("pdus");
            arrayOfSmsMessage = new SmsMessage[arrayOfObject.length];
        }
        label494:
        label627:
        label628:
        for (int i = 0;; i++)
        {
            String str10;
            String str11;
            String str12;
            String str13;
            String str14;
            String str15;
            String str16;
            String str17;
            if (i >= arrayOfSmsMessage.length)
            {
                if (bool) {
                    break label494;
                }
                Intent localIntent1 = new Intent(paramContext, SMSHandlerService.class);
                Bundle localBundle2 = new Bundle();
                localBundle2.putString("FROM", str1);
                localBundle2.putString("MESSAGE", str2);
                localIntent1.putExtras(localBundle2);
                paramContext.startService(localIntent1);
                SharedPreferences localSharedPreferences = GF.getSavePref(paramContext);
            }
        }
    }
}
```

הרבה APK-ים משתמשים בשיטות Obfuscation שונות להסתרת המטרות האמיתיות של ה-class-ים שלהם. בדרך כלל זה כולל פתיחה של המחרוזות רק בזמן ריצה וכן שינוי של ה-class-ים לשמות כגון a, b, c וכדומה. למזלנו, כאן לא ננקטה גישה זו ולכן קל לנו מאד להבין את מטרות ה-class-ים השונים.

לעיתים קוד הפלט של dex2jar יוצא מכוער או אפילו לא מדוייק, ולעיתים התרגום לא מצליח (ואז צריך להתחיל לתרגם ידנית קוד SMALI לקוד Java).

אם כן, מתודת ה-onReceive תקבל את ה-SMS בתוך שדה ה-extra של ה-Intent. ניתן לראות כי עבור כל הודעה יישלח Intent נוסף - הפעם אל class בשם SMSHandlerService, עם שני שדות: FROM ו-MESSAGE.

```
protected void onHandleIntent(Intent paramIntent)
{
    Log("--onHandleIntent--");
    setupAnalytics();
    this.pref = GF.getSavePref(this);
    loadSettings();
    Log("Get bundle passed from SMS Receiver");
    Bundle localBundle1 = paramIntent.getExtras();
    this.from = localBundle1.getString("FROM");
    this.message = localBundle1.getString("MESSAGE");
    Log(2, "From: " + this.from + "\n" + " - Message: " + this.message);
    if ((this.from == null) || (this.message == null)) {
        Log("from or message is null");
    }
    String str;
    do
    {
        return;
        if ((this.from.contains("@") || (this.message.contains("@"))))
        {
            Intent localIntent = new Intent(this, SMSEmailHandlerService.class);
            Bundle localBundle2 = new Bundle();
            localBundle2.putString("FROM", this.from);
            localBundle2.putString("MESSAGE", this.message);
            localIntent.putExtras(localBundle2);
            startService(localIntent);
            return;
        }
        str = GF.trimText(this.message);
        if (str.startsWith("wmd open")) {
            openApp(this.message);
        }
        this.ringAttWord = GF.trimText(this.ringAttWord);
        this.gpsAttWord = GF.trimText(this.gpsAttWord);
        this.camAttWordBack = GF.trimText(this.camAttWordBack);
        this.camAttWordFront = GF.trimText(this.camAttWordFront);
        this.lockAttentionWord = GF.trimText(this.lockAttentionWord);
        this.unlockAttentionWord = GF.trimText(this.unlockAttentionWord);
        this.wipeAttentionWord = GF.trimText(this.wipeAttentionWord);
    } while (!containsAttentionWord(str));
    if (this.whtblkListEnabled.booleanValue())
    {
        if (whtblkCheck(this.whtblkWhiteEnabled))
        {
            Log("we can continue");
            checkAttWord(str);
            return;
        }
        Log("rejected we can't continue");
        return;
    }
    checkAttWord(str);
}
}
```

נשים לב לקוד הברור שיצא - במידה והתקבל Intent מייל (מזוהה על ידי סימן ה-@), אז מטפל בו ה-SMSEmailHandlerService. אחרת, הקוד הנוכחי יטפל בו. אם נתעלם מפקודת wmd open, נשים לב שדי הרבה member-ים מאותחלים, ולאחר מכן בודקים האם ההודעה מכילה את אחת מהמילים הללו



בפונקציה הפנימית containsAttentionWord). לאחר מכן קוראים אל checkAttWord. נבחן את המתודה הזו (הקוד לא מלא):

```
private void checkAttWord(String paramString)
{
    Log("checkAttWord");
    if (paramString.contains("wmdprounlock"))
    {
        Log("Version changed to pro");
        GF.getSavePref(this).edit().putInt("version", 1).putInt("nag_count", 0).commit();
        return;
    }
    if (paramString.startsWith("wmdinstalled"))
    {
        Log("Message equals respond string");
        Analytics.Event(tracker, "feature_used", "sms_action", "installed_request");
        GF.sendMessage(this, this.from, getString(2131165286));
        return;
    }
    if (paramString.startsWith(this.ringAttWord))
    {
        Log("Message equals ring attention word");
        ringFeature(this.from);
        return;
    }
    if (paramString.startsWith(this.gpsAttWord))
    {
        Log("Message equals GPS attention word");
        gpsFeature(this.from);
        return;
    }
    if (paramString.startsWith(this.camAttWordBack))
    {
        Log("Message equals camera back attention word");
        cameraBackFeature(this.from);
        return;
    }
    if (paramString.startsWith(this.camAttWordFront))
    {
        Log("Message equals camera front attention word");
        cameraFrontFeature(this.from);
        return;
    }
    if (paramString.startsWith(this.lockAttentionWord))
    {
        Log("Message equals remote lock attention word");
        lockFeature(this.from, paramString);
        return;
    }
    if (paramString.startsWith(this.unlockAttentionWord))
    {
        Log("Message equals unlock attention word");
        unlockFeature(this.from);
        return;
    }
}
```

אז מה יש לנו?

- SMS ניקלט על ידי SMSReceiver.
- SMSReceiver שולח Intent אל SMSMessageHandler עם שדות ה-FROM ו-MESSAGE.
- מתודת onHandleIntent מתבצעת, וכתלות בתוכן ההודעה עלולה להתבצע אחת מהפעולות הבאות:
 - עדכון האפליקציה לגרסת Pro (ההודעה "wmdupgrade").
 - בדיקה האם האפליקציה מותקנת (ההודעה "wmdinstalled").
 - צלצול.
 - שליפת נתוני GPS.
 - לקיחת תמונה (באופן שקט) מתוך המצלמה הקדמית או האחורית (דורש גרסת Pro).
 - נעילה ושחרור של הטלפון (דורש הפיכת האפליקציה ל-Device administrator).
 - ביצוע Wipe למכשיר (דורש הפיכת האפליקציה ל-Device administrator).
- ההודעות שגורמות להפעלות (להוציא את השתיים הראשונות שהן hard-coded) קונפיגורביליות, אך מקבלות ערכים default-יים:
 - צלצול: "WMD Ring".
 - שליפת מיקום: "WMD GPS".
 - תמונה: "WMD Camera Front" ו-"WMD Camera Back".
 - נעילה ושחרור: "WMD Lock" ו-"WMD Unlock".
 - ביצוע Wipe למכשיר: "WMD Wipe".
- את הערכים הללו ניתן לשלוף בקלות מתוך strings.xml (שמכיל את ה-String resources של האפליקציה).

תובנות נוספות:

- האפליקציה לא מתריעה לאחר ההתקנה על כך שהערכים הם default-יים.
- שדרוג ל-Pro אמור לעלות כסף. עם זאת, נראה שניתן לשדרג בחינם על ידי שליחת "wmdupgrade".
- חלק מהפיצ'רים באפליקציה מאופשרים רק לאחר שדרוג ל-Pro. תוקף יכול לשדרג מרחוק כמובן.
- חלק מהפיצ'רים דורשים מהאפליקציה להיות Device administrator, מה שלא מתבצע כברירת מחדל.
- אם נסכם, עבור התקנה "רגילה" של האפליקציה ללא שינויים, תוקף יכול לבצע (בסיכוי טוב):
 - שדרוג ל-Pro.
 - צילום שקט מהמצלמות. התמונה עולה באופן שקט ואוטומאטי לשרת מרוחק, ולינק נשלח לתוקף.
 - שליפת נתוני GPS. הנתונים נשלחים באופן שקט כ-SMS החוצה לתוקף.
- נציין כי יש אפשרויות של האפליקציה לביצוע whitelist, אך היא לא דולקת כברירת מחדל.

משטחי תקיפה באפליקציות - Android חלק א'

www.DigitalWhisper.co.il



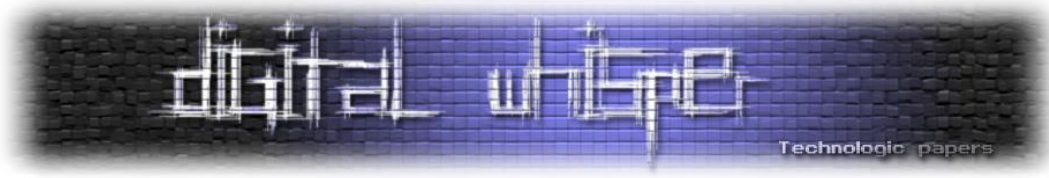
בעיות נוספות

תחת אנדרואיד, רכיבים (ו-Service-ים ביניהם) יכולים להיות exported או לא להיות exported. כאשר רכיב הוא exported אז הוא יכול לקבל Intent-ים מאפליקציות אחרות, וכאשר הוא לא exported אז הוא לא. ניתן לראות כי SMSHandlerService הוא exported, ולכן אפליקציה סוררת (ללא הרשאות קריאת SMS-ים, ללא הרשאות מצלמה וכדומה) יכולה לשלוח Intent שייתפס על ידי ה-SMSHandlerService ויבצע פעולות בשמה.

מסקנות

1. בתחילת מאמר זה סקרנו (באופן גס מאד) את היכולות של אנדרואיד. לאחר מכן, התמקדנו במשטח תקיפה מבוסס SMS-ים, והצגנו כיצד אפליקציה תמימה למראה יכולה לשמש ככלי ריגול לכל דבר.
2. כמו רוב החולשות האפליקטיביות באנדרואיד, הבעיה נבעה ממתכנת שסומך לחלוטין על הקלט הנכנס, בצירוף חוסר הבנה על כך שניתן לבצע Reversing לאפליקציה עצמה. נפוץ מאד לראות סיסמאות hard-coded באפליקציות לאנדרואיד, והמצב הנוכחי הוא שאין תהליך מסודר שמבצע Review על קוד של אפליקציה.
3. אפליקציה עם ערכים דיפולטיים שלא מתריעה בפני המשתמשים שלה על כך שיש לעדכן את הערכים הללו היא אפליקציה בעייתית (באופן דומה לקוד הסודי לתא הקולי הסלולרי או הסימא לראוטרם הביתיים של חלק גדול מאיתנו).

אף על פי שסדרת המאמרים תופץ גם בעברית, אני מזמין את הקורא השקדן לקרוא את הפוסט המקורי בבלוג שלי, בכתובת: <http://securitygodmode.blogspot.com>



מבוא לאסמבלי

מאת אופיר בק

הקדמה

בסדרת המאמרים הקרובה, אנחנו הולכים ללמוד על השפה אסמבלי, על השימוש בה ואף נבנה בעזרה מספר תוכנות קטנות בשביל הכיף. במבוא נסביר קצת על אסמבלי, איך להריץ אותו ונראה את הקוד הראשון שלנו. אך בשלב זה אסביר בעיקר על מבנה המחשב, מידע אשר הכרחי בשביל לדעת אסמבלי.

מהי שפת אסמבלי?

"שפת סף אשר נקראת גם אסמבלי היא שפת התכנות הבסיסית ביותר והקרובה ביותר לשפת מכונה." - ויקיפדיה.
בשונה מהרבה שפות אחרות שניתן ללמוד, כדי ללמוד אסמבלי צריך לדעת קצת יותר על איך המחשב עובד, אז אנחנו נתמקד בדרישות הידע הכי פשוטות בחלק הזה.

התפתחות שפות התכנות

בתחילה כדי להורות למחשב לבצע חישובים, מפתחים נדרשו לרשום את ההוראות בשפה הבינארית, שפת המכונה, עליה נרחיב בהמשך. כלומר קודים היו נראים ככה:

```
0101 1100 1110 1111 1110 0110 0011...
```

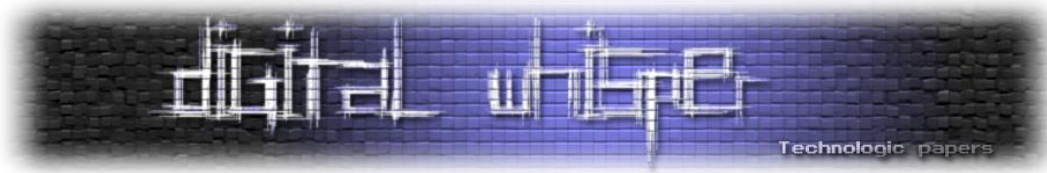
הבעיות הראשיות הן שאי אפשר לקרוא את הקוד ולהבין אותו, וקשה למצוא טעויות.

כאן בדיוק נכנסת שפת האסמבלי. הפקודות של שפת האסמבלי הן באנגלית, קל להבין את משמעותן ולמצוא טעויות בקלות רבה יותר. לכן נקראת שפת האסמבלי שפת סף, כי היא על סף להיות שפת מכונה. הפקודות של השפה דרושות הכרה עמוקה עם המעבד כדי למנוע טעויות קריטיות שעלולות לפגוע במחשב, אך אנו נשתמש ב-Emulator שידמה מערכת הפעלה ישנה שתחסוך לנו את הדאגה.

מעל הרמה הזאת יושבות שפות יותר פשוטות לקריאה ולכתיבה, בנוסף למציאת טעויות, והן נקראות שפות עיליות, וביניהן ניתן למצוא שפות כמו C# ו-Java. השפות האלו בנויות עם מנגנונים שמונעים טעויות של המעבד ולא מריצים את הקוד במידה והם מוצאים טעויות. הן בעצם 'מסתירות' מהתוכניתן את המעבד.

מבוא לאסמבלי

www.DigitalWhisper.co.il



למה לא ללמוד אסמבלי?

כן, אני לא הולך להסתיר מכם את הקשיים שיש בלימוד אסמבלי:

1. מאז שאסמבלי פותחה (בשנת 1949) העולם התקדם ויש שפות תוכנה מודרניות
2. אסמבלי היא שפה שקשה ללמוד
3. מסובך וארוך לכתוב קודים בשפת אסמבלי
4. קשה לבצע דיבאג (למצוא שגיאות ולנפות אותן) באסמבלי, לעומת השפות העיליות

למה כן ללמוד אסמבלי?

בכל זאת, יש יתרונות בלימוד השפה וידיעתה:

1. ניצול מיטבי של משאבי המעבד
2. עבודה מול החומרה של המחשב
3. גודל קובץ קטן מאוד ביחס לשפות אחרות (ראיתם פעם את התיקייה שנוצרת עבור כל קובץ C# קצרצר?)
4. הבנה עמוקה של אופן פעולת המחשב (אין דברים שמוסתרים מהתוכניתן)
5. רכישת מימוניות של סדר וארגון

שיטות ספירה

בני אדם נוהגים לספור בשיטה של 10 ספרות, מ-0 ועד 9, שכאשר אנו מוסיפים ספרה משמאל לספרה הראשונה אנו מגדילים את הערך של הספרה השמאלית פי 10 והספרה הבאה אחריה תהיה גדולה פי 100 שזה בעצם 10^2 וכן הלאה, לדוגמה 16 זה בעצם $1 \times 10 + 6$, והמספר 458 הוא בעצם $4 \times 10^2 + 5 \times 10 + 8$.

לעומת זאת, המחשב סופר בבסיס 2, כלומר, הוא משתמש רק בערכים 1 ו-0. גם פה ייצוג המספרים הגדולים מכמות הספרות נעשים באמצעות הוספה של מספר משמאל למספר השמאלי ביותר. לדוגמה כדי לייצג את המספר שמוכר לנו כ-2 אנו נרשום 10 בבסיס הבינארי (2), מכיון ש $2 = 1 \times 2 + 0$.

נהוג לרשום את הקוד הבינארי בבלוקים של ארבעה תווים כדי להקל עלינו לקרוא אותו ולרשום בסיום b. כך לדוג' אנו נרשום את המספר 2 בבסיס 2 בתור 0010b, ואת המספר 15 בתור 1111b. כל תו בינארי נקרא סיבית - סיפרה בינארית.

עם זאת, כדי להקל על התוכניתנים, המחשב יודע לקרוא גם בבסיס הקסדצימלי (16). בבסיס הזה אנו משתמשים בספרות ובנוסף להן בתווים A עד F, כדי להשלים את ששת הספרות החסרות. בבסיס הזה נהוג לרשום את המספר עם האות h בסופו או 0x בתחילה.



כדי לתת למחשב את המספר הדיצימלי, פשוט לא מוסיפים אף סימן בתחילה או בסוף, וכך הוא מניח שמדובר בשיטת הספירה שאנו משתמשים.

שימו לב! ניתן להמיר בין הבסיסים הקסדצימלי והבינארי בקלות, מכיוון שכל 4 ביטים מייצגים תו הקסה אחד, כך ש $0001b=1h$ ו- $1111b=Fh$.

כאשר אנו עובדים עם בית אחד, הערכים נעים בין 0 ל-255, אך כאשר אנו רוצים לייצג מספר שלילי, הם נעים בין 128- ל-127. המחשב לא באמת מבדיל בין המספרים, אך בעזרת פעולות מסויימות אנו מגדירים לו כיצד להתייחס אליהם.

השיטה להתייחסות לקוד בינארי כשלילי נקראת "משלים ל-2". כדי לייצג מספר בשיטה הזו, אנו מציגים אותו בפן החיובי, הופכים את כל הביטים ומוסיפים אחד. כלומר, המספר 1- בשיטה הזו יתואר בצורה הבאה:

(1) תחילה נביע את המספר 1 באופן הבא: 0000 0001.

(2) נהפוך את כל הביטים: 1111 1110.

(3) נוסיף 1: 1111 1111.

וככה אנו מביעים את המספר 1-.

בשיטה הזו כל מספר שהביט השמאלי שלו הוא 1, הוא שלילי.

מבנה המחשב

כמו שהזכרנו קודם, אסמבלי עובד בצורה הדוקה מול החומרה של המחשב, ולכן יש מספר יתרונות בהכרת המחשב והמבנה שלו:

1. אופטימיזציה של הקוד
2. הקטנת גודל הזיכרון של חלקים בקוד (בהמשך נגלה למה זה משמעותי)
3. מאפשר דיבאג יותר טוב של הקוד.

אנחנו נעבוד עם מעבד 8086 של אינטל, שיצא בשנת 1978 והיה הראשון בסדרה 80x86. הסיבה שעבודה איתו היא רלוונטית היא בגלל עיקרון שנקרא "תאימות לאחור", לפיו גם מעבדים חדשים יותר של אינטל מסוגלים לעבוד עם הקוד שנועד למעבד ישן יותר.

המעבד הנ"ל בנוי בארכיטקטורה שנקראת "ארכיטקטורת פון ניומן", על שם היוצר של המעבד, ג'ון פון ניומן. לפי הארכיטקטורה, ישנם שלושה 'פסים' במערכת, שניתן לפנות רק אל אחד מהם בכל פעם. הפסים האלו הם הבקרה, המענים, והמידע. והם מחוברים בנפרד לרכיבי הקלט והפלט, למעבד ולזיכרון.



פס הבקרה במחשב אחראי לומר למעבד האם אנו מבצעים פעולה של קריאה או של כתיבה, והאם לפנות לקלט-פלט או לזיכרון. הערכים של הקריאה והכתיבה הם בד"כ על 1, וכאשר read=0 מתבצעת קריאה, וכאשר write=0 מתבצעת כתיבה.

פס המענים מודיע לאיזו כתובת בזיכרון המעבד מכוון. בזיכרון כל מקום הוא בגודל של בית (8 ביט), והוא מסודר כך שאם נכניס אליו את הערך 1234h הוא יסודר כך שקודם כל יישמר בזיכרון 34h ורק לאחריו 12h. שיטה זו קרויה little-endian.

פס הנתונים מעתיק נתונים ממקום למקום. מעבד עם פס נתונים רחב יותר יכול להעביר מידע מהר יותר. פס הנתונים של המעבד 8086 מסוגל להעתיק 16 ביטים בהעתקה אחת.

צריך לשים לב שמבחינה עקרונית, זיכרון לא יכול להיות ריק, לעיתים הוא עם מידע שאנו הכנסנו לו, ואז הוא שמיש מבחינתנו, אך לעיתים מדובר במידע זבל שאינו אמין, וקריאה שלו עלולה לגרום לבעיות בהמשך.

בנוסף לכך, מכיוון שלמעבד ה-8086 יש מרחב כתובת של 20 ביטים, הוא משתמש בשיטה של segment (קטע בזיכרון) ו-offset (מיקום בקטע) ופונה לזיכרון במיקום שבנוי בתור segment: offset.

המעבד

המעבד 8086 בנוי ממספר חלקים, ואנו נפרט עבור אלו שחשובים לתכנות האסמבלי שלנו:

- **רגיסטרים - Registers**: הרגיסטרים מתחלקים לכמה סוגים, וכשנתחיל לכתוב תוכניות אנו גם נפרט עליהם.
- **יחידה אריתמטית לוגית - Arithmetic Logic Unit**: היחידה האריתמטית לוגית אחראית על ביצוע פעולות מתמטיות ולוגיות עבור המעבד.
- יחידת בקרה.
- **יחידת קלט/פלט - I/O Ports**: ניתן להשתמש ביחידה הזו כדי לקבל קלט מהמשתמש.
- **שעון - Timer**: מאפשר שימוש בפונקציות זמן מסוגים שונים.

אוגרים

האוגרים, או כפי שנכנה אותם בד"כ, רגיסטרים, הם רכיבי חומרה שצמודים למעבד, והם מאפשרים לנו לבצע מגוון פעולות. בשונה מהזיכרון, אל האוגרים המעבד יכול לפנות באופן ישיר, וללא שום המתנה, בשל מיקומם הפיזי. הכמות והגודל של הרגיסטרים משתנה בין דורות של מעבדים, אבל אנחנו נשתמש ברגיסטרים של מעבד ה-8086.

לרגיסטרים שונים יש שמות שונים ומטרות רשמיות שונות, אך בעיקרון, את מרבית הפעולות רובם יכולים לבצע:

- **AX** - Accumulator: הרגיסטר המתמטי. הוא משמש לרוב הפעילויות האריתמטיות והלוגיות, והוא בד"כ יעיל יותר בביצוען.
- **BX** - Base: בדרך כלל משמש לשמירת כתובות בזיכרון, מכיוון שהוא בין היחידים שיש להם גישה לזיכרון.
- **CX** - Counter: רגיסטר שהוא ייעודי לספירה, עבור לולאות, כמות תווים בקובץ או במחרוזת וכדומה.
- **DX** - Data: שומר מידע עבור גישות מיוחדות לזיכרון או עבור פעולות מיוחדות, ועבור חלק מהפעולות האריתמטיות הוא משמש כרגיסטר נוסף.
- **SI** ו-**DI** - Destination Index | Source Index: משמשים לגישה לזיכרון עם BX, ולאחסון נוסף ברגיסטרים.
- **BP** - Base Pointer: משמש לגישה לזיכרון של הסגמנט של המחסנית (Stack Segment).
- **SP** - Stack Pointer: שומר את המיקום הנוכחי במחסנית. בד"כ לא נשנה את הערך שלו ידנית, אבל זה עשוי להימצא יעיל לעיתים.

כדי לאפשר גישה לקטעי זיכרון יותר קטנים מ-16 ביט, אפשר לפנות רק לחצי רגיסטר, כאשר עבור AX, BX, CX, DX אנו נהפוך את ה-X ל-L עבור החצי התחתון של הרגיסטר ו-H עבור החלק העליון, מלשון High ו-Low. עבור הרגיסטרים SI ו-DI ניתן להוסיף בסוף L או H, מבלי למחוק את האות I.

בנוסף לרגיסטרים הכלליים שהזכרנו קודם, ישנם גם רגיסטרים מיוחדים שנקראים רגיסטרי מקטע, או Segment Registers. ישנם ארבע רגיסטרים כאלה:

1. **DS**, או Data Segment, מצביע על המקום בזיכרון בו שמורים המשתנים שלנו.
2. **CS**, או Code Segment, מצביע על המקום בו שמור הקוד שלנו בזיכרון המחשב.
3. **SS**, או Stack Segment, מצביע על איזור המחסנית בזיכרון המחשב.
4. **ES**, או Extra Segment, מצביע על אזור נוסף בזיכרון, שבמידה ואנו חורגים ממגבלות הקוד של הסגמנטים (64KB כל אחד) אנו יכולים להשתמש בו.



הרגיסטרים האחרונים הם IP, או ה-Instruction Pointer, שמצביע על המיקום בזיכרון של השורה הבאה להרצה, ו-FLAGS, שחשוב מאוד עבור ביצוע תנאים לוגיים ולולאות.

הכנות לאסמבלי

אנחנו נשתמש ב-notepad++ החינמי בתור סביבת העבודה שלנו, ובשביל להתאים את זה לשפת האסמבלי, ניגשים לכפתור Language בתפריט העליון ותחת האות A בוחרים באפשרות Assembly.

בנוסף לכך, כדי להריץ את הקבצים אנו נשתמש באימולטור DOSBox, על השימוש בו נסביר בהמשך. תוספת אחרונה שלה אנו נזדקק היא TASM ו-TLINK. TASM (Turbo Assembler) הוא האסמבלר הבסיסי שבו נשתמש, והוא לא מורכב כמו רבים מהאחרים, כמו NASM, עליהם אולי ארחיב בהמשך. אנו משתמשים בהם (ובקבצים נוספים) כדי להפוך את קוד האסמבלי שלנו לקובץ מסוג EXE (Executable), שנוכל להריץ ב-DOSBox.

קישור להורדה מסודרת:

<https://drive.google.com/open?id=0B7fBWISzrcHTam96NjB6RmtpdzQ>

חלצו את הקבצים, התקינו את DOSBox ואת Notepad++, והעבירו את התיקיה BIN לכונן C.

Base.asm

פתחו את הקובץ base.asm שנמצא בתיקייה BIN, בעזרת notepad++. הוא הבסיס הקבוע יחסית לכל קבצי האסמבלי שנכתוב, ועכשיו נסביר כל שורה ממנו. שימו לב, כל הפקודות באסמבלי הן לא Case Sensitive, כלומר, אפשר לרשום גם באותיות גדולות וגם באותיות קטנות. עם זאת, נהוג לרשום באותיות גדולות את ההוראות למחדר (האסמבלר) ובאותיות קטנות את הקוד עצמו:

- **IDEAL** - זאת הוראה לאסמבלר שלנו, TASM, שמודיע לו שאנו עובדים במצב אידיאלי. אנו לא נשתמש במצבים אחרים במסגרת הזו, אבל תוכלו למצוא עוד הרבה אופציות באינטרנט.
- **MODEL SMALL** - אנו נשתמש רק במודל הזה, בו יש segment אחד של קוד, ו-segment אחד של מידע (בנוסף לאחד של המחסנית, בו ניגע בהמשך).
- **STACK 100h** - הצהרה על גודל המחסנית. כל מקום במחסנית הוא בית אחד, כלומר שמונה ביטים, שהם 2 תווים הקסדצימליים. אז בכתיבת הפקודה הזו, בעצם הצהרנו על 50 מקומות במחסנית. שימו לב שתמיד יש צורך להצהיר על גודל המחסנית, גם אם אתם לא מתכננים להשתמש בה.
- **DATASEG** - זה בעצם אזור המידע בקוד שלנו. כאן אנו מצהירים על המשתנים שאנו רוצים לשמור בזיכרון. תכף נדבר גם על איך עושים את זה.

מבוא לאסמבלי

www.DigitalWhisper.co.il



- **CODESEG** - אזור הקוד בזיכרון שלנו. פה אנחנו רושמים את הקוד עצמו.
- **start:** - הצהרה על תחילת הקוד. השם לא באמת משנה, אבל נהוג לקרוא לו start. הוא חייב להתאים גם לתגית הסיום END, שמופיע בסוף הקוד בתור END start.
- **mov ax, @data** - אנחנו בעצם מעבירים לרגיסטר ax את המיקום שבו מתחילה שמירת הנתונים שלנו. צריך לבצע את זה בתחילת כל קובץ אסמבלי. השימוש בסימן @- נועד כדי לקרוא למילה השמורה data, ולא למשתנה בשם הזה אם היינו מחליטים לבנות אחד שכזה.
- **mov ds, ax** - מעבירים לרגיסטר שאחראי על המשתנים את המיקום שבו הם שמורים.
- **exit:** - מלבד התגית הראשונה, שלה קראנו start, האסמבלי משתמש בשיטה שקוראים לה Labeling, ובעצם מתבססת על כך שנקרא לכל קטע קוד תחת איזושהי תגית, כדי שנוכל לקפוץ ביניהם בהמשך. כרגע לא נעמיק במשמעות השורות הבאות בקוד, מלבד END start, אותה כבר הסברנו קודם.

בקובץ יש שיטה של אינדנטציה, או בעברית - הזחה. כל מה שנמצא תחת DATASEG נמצא TAB אחד פנימה יותר, וכן כל מה שנמצא תחת התגיות הפנימיות של שם-CODESEG. האסמבלי לא קורא את ההזחות בכל מקרה, אך הן מקלות על המשתמש את ההבנה של הקוד.

DOSBox

המקור של האימולטור DOSBox הוא בעצם במערכת ההפעלה DOS שהייתה פעילה גם לאחר יציאת Windows, בשל מדיניות ה-"תאימות לאחור" שהזכרנו קודם לכן.

האימולטור נדרש מכיוון שלמרות מערכת ההגנה של Windows, שהייתה מונעת מאיתנו לפגוע במחשב בעזרת אסמבלי בדרך כלל, הוחלט שלא לתמוך באסמבלי 16 ביט, מה שמשאיר אותנו עם מערכת DOS חביבה. כשפותחים את ה-DOSBox הוא מוביל אותנו לכונן Z, שהוא כונן וירטואלי, אך כדי להקל את השימוש אנחנו נשתמש באוסף הפקודות הבא:

- `mount c: c:\` - בקשת מעבר לכונן C, בו נמצאת התיקייה BIN.
- `c:` - מעבר לכונן C.
- `cycles = max` - העלת מספר סיבובי המעבד למקסימום (אל תדאגו, זה רק המעבד המדומה, ומרבית המעבדים היום יעמדו בזה בקלות רבה).
- `cd bin` - מעבר לתיקייה bin.

זהו! עכשיו אנו נמצאים בתיקייה בה נשמור את קבצי העבודה. אם תרצו פירוט על פקודות נוספות או על אלו שהזכרנו, תוכלו למצוא את זה באינטרנט.

מבוא לאסמבלי

www.DigitalWhisper.co.il



השלב הבא הוא חשוב לא פחות - ההפיכה של קובץ ה-asm שלנו לקובץ שהדוס יוכל להריץ. את זה אנו נעשה בשני שלבים:

1. `tasm /zi base.asm` - או כל שם קובץ אחר שנבחר. זה בעצם אסמבלר, והוא מתרגם את שפת האסמבלי שלנו לשפת מכונה, כלומר קובץ עם סיומת `.obj`.

2. `tlink /v base.obj` - או כל שם קובץ אחר שנבחר. את הפעולה הזאת מבצע לינקר שיודע לאחד עבורנו כמה קבצים לתוכנית אחת, אך מכיוון ששלנו היא רק קובץ אחד בכל מקרה, השימוש הוא פשוט יחסית. עכשיו נוצר לנו קובץ `.exe`.

עכשיו אנו מגיעים לצומת, מכיוון שיש שתי דרכים להריץ את הקובץ, האחת עם Turbo Debugger, שייתן לנו את האופציה לעקוב אחרי פעולת התוכנית, ואחת רגילה.

כדי להריץ תחת הדיבאגר, רושמים:

```
td base
```

וזה יפתח את הקובץ בצורה המתאימה. אחרת, רושמים רק את שם בקובץ, במקרה שלנו:

```
base
```

התהליך נשמע מתיש בשביל כל הרצה של קוד, נכון? לכן יש לנו את הקישור הבא:

<https://drive.google.com/open?id=0B7fBWwSzcHTbzh3eIJDZxNjYzg>

בקישור תמצאו RAR שמכיל שני קבצי `batch`, כלומר, בעלי סיומת `.bat`. תעתיקו אותם אל התיקייה `bin`, ומעכשיו, כשתרצו להריץ קובץ בלי הדיבאגר, תוכלו לרשום `run base`, ובמידה ותרצו להשתמש בדיבאגר, תרשמו `rund base`, ולאחר בדיקת טעויות הקובץ ירוץ אם הוא לא נתקל בשגיאה.

IP-FLAGS

כבר הזכרנו את שניהם כשדיברנו על מבנה המחשב, אבל הפעם אנחנו ניכנס עמוק יותר לתוך השימוש של כל אחד מהם.

IP, או Instruction Pointer הוא המצביע על הפקודה הבאה בקוד. הפקודות מגיעות בגדלים שונים, בד"כ בין בית אחד לשלושה, וה-IP אמור לוודא שהמעבד יקלוט את הפקודות כראוי, ולא יחבר בין כמה ביטים מפקודה אחת וכמה מאחרת, ויקבל הבנה שגויה של הפקודה שלנו.

FLAGS - בשונה משאר הרגיסטרים, ב-**FLAGS** יותר קשה לעשות שימוש חופשי, ובפועל יש שם שימוש רק בחלק מהביטים לצורך תנאים וכד'.

הביטים השונים נותנים לנו מידע על מצב המעבד לאחר הפקודה האחרונה שהרצנו, ואנחנו נתמקד בארבעה מהדגלים, הנקראים דגלי בקרה:

1. **Zero Flag** - הערך של דגל הזה הופך ל-1 במידה ולאחר הרצת הפקודה האחרונה האופרנד התאפס. האופרנד הוא בעצם מקום בזיכרון, או רגיסטר, שבמקרה הזה הם אלו שקיבלו את תוצאת הפעולה. לדוגמה, אם נחבר 1 ו-1 נקבל 0, ואז הדגל יקבל את הערך 1. גם חיבור של המספר 255 עם המספר 1 יהפוך לאפס, מכיוון שהוא חוצה את גבול המספרים שניתן לייצג בשמונה ביטים (זה לא יקרה לרגיסטרים של 16 ביט לדוגמה).
2. **Carry Flag** - הדגל הזה מקבל את הערך 1 אם תוצאת הפעולה חורגת מתחום המספרים הרגילים (ללא התייחסות לשליליות). עבור 8 ביט החריגה היא מהתחום של 0-255, ועבור 16 התחום הוא 65535-0
3. **Overflow Flag** - הדגל הזה מקבל את הערך 1 אם תוצאת הפעולה חורגת מתחום המספרים המסומנים, שעבור 8 ביט החריגה היא מ-128 עד +127. עבור 16 ביט התחום הוא בין 32768- לבין +32767.
4. **Sign Flag** - דגל הסימן. מקבל את הערך 1 כאשר הביט השמאלי בתוצאה הוא 1 (כמו שהזכרנו קודם, במקרה כזה המספר הוא שלילי לפי שיטת המשלים ל-2).

הגדרת משתנים

בתכנות נהוג להשתמש במשתנים כדי לשמור מספרים, והמקור לכך הוא באסמבלי, שבו משתמשים במשתנים כדי לחסוך את המבט לזיכרון בכל פעם שרוצים להשתמש בערך שמור. כמו שכבר הזכרנו קודם, את המשתנים אנו יוצרים ב-DATASEG של הקובץ שלנו באופן הבא:

Name size value - כלומר כדי ליצור משתנה בשם age עם בגודל של בית אחד, עם הערך ההתחלתי 16, אנו נרשום ב-DATASEG:

```
age db 16
```

הפירוש של DB הוא Define Byte, אך כדי ליצור משתנים בגדלים אחרים, צריך להחליף את הצירוף. ניתן ליצור משתנים בגודל מילה (WORD), שהיא שני בתים בעזרת הצירוף DW, וליצור משתנה בגודל מילה כפולה (DWORD) בעזרת הצירוף DD. כדי לפנות למשתנים בקוד עצמו, אנו עוטפים אותם בסוגריים מרובעים משני הצדדים.

לדוגמה, כדי להעביר למשתנה age את הערך 17, אנו נשתמש בפקודה mov, שמזיזה ערכים בין מיקומים, באופן הבא:

```
mov [age], 5
```



ולא:

```
mov 5, [age]
```

הסיבה לכך היא שלפקודות באסמבלי יש עד שני אופרנדים, ועבור הפקודה mov, בשונה מרוב הפקודות האחרות, האופרנד הראשון שאנו מכניסים, במקרה הזה mov הוא אופרנד היעד, שאליו מועתק הערך מהאופרנד השני, שנקרא גם אופרנד המקור.

כדי להגדיר מחרוזת, כלומר אוסף תווים אנו יכולים לרשום אותם החל מהתו הראשון באופן הבא:

```
string db 'HELLO'
```

במקרה כזה כל אחד מהתווים יישמר בבית נפרד, אחד אחרי השני, ברצף הנכון. באסמבלי יש גם הגדרה של מערכים, אך היא שונה במקצת:

```
ArrayName SizeOfElement N dup(?)
```

- ArrayName - שם המערך
- SizeOfElement - גודל כל תא (...DB, DW)
- N - מספר הפעמים לביצוע ההעתקה של הערך (או רצף הערכים) בסוגריים.
- dup - פקודת העתקה (duplicate).

גם באסמבלי יש אינדקסים במערך, שעוזרים לנו למצוא את המיקום של תא מסוים. האינדקס הראשון הוא 0. כדי לקבוע את כל הקפיצות בין אינדקסים, צריך לספור את גודל התא, מכיוון שהזיכרון בנוי מבתים, ולכן במקרה של מערך מסוג WORD לדוגמה, האינדקס של התא השני יהיה 2 ולא אחד, מכיוון ש WORD מבוטא בגודל של שני בתים.



לסיכום

למדנו את הבסיס לאסמבלי, לדוגמה שיטות ספירה (בינארית והקסדצימלית), מבנה המחשב, וגם יצירת משתנים ומערכים. בפרק הבא נעסוק בפקודות בסיסיות ונכתוב קודים בסיסיים באסמבלי.

על המחבר

שמי אופיר בק, בן 16 מפתח תקווה. אני לומד בתכנית גבהים של מטה הסייבר הצה"לי וב-C Security, לאחר שסיימתי את לימודי המתמטיקה והאנגלית בכיתה י'. קשה למצוא חומר מעודכן בעברית, ולאחר שהמגזין הזה היווה עבורי מקור מידע נגיש, רציתי לתרום חזרה. זה המאמר הראשון שלי במגזין ובכלל. ניתן גם ליצור איתי קשר בכתובת ophiri99@gmail.com.

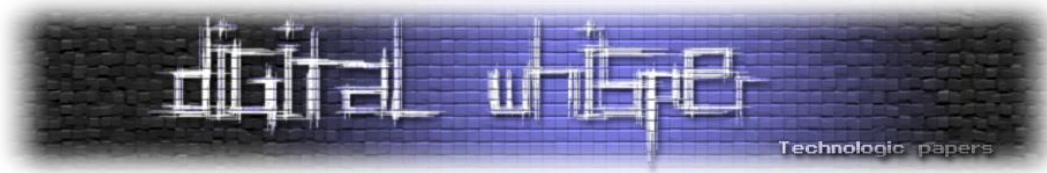
קישורים לקריאה נוספת

- טורבו דיבאגר:

https://en.wikipedia.org/wiki/Borland_Turbo_Debugger

- מבנה המחשב:

<http://www.rup.co.il/sites/default/files/%20%D7%9E%D7%91%D7%A0%D7%94%20%D7%94%D7%9E%D7%97%D7%A9%D7%91.docx>



דברי סיכום

בזאת אנחנו סוגרים את הגליון ה-73 של Digital Whisper, אנו מאוד מקווים כי נהנתם מהגליון והכי חשוב- למדתם ממנו. כמו בגליונות הקודמים, גם הפעם הושקעו הרבה מחשבה, יצירתיות, עבודה קשה ושעות שינה אבודות כדי להביא לכם את הגליון.

אנחנו מחפשים כתבים, מאיירים, עורכים ואנשים המעוניינים לעזור ולתרום לגליונות הבאים. אם אתם רוצים לעזור לנו ולהשתתף במגזין Digital Whisper - צרו קשר!

ניתן לשלוח כתבות וכל פניה אחרת דרך עמוד "צור קשר" באתר שלנו, או לשלוח אותן לדואר האלקטרוני שלנו, בכתובת editor@digitalwhisper.co.il.

על מנת לקרוא גליונות נוספים, ליצור עימנו קשר ולהצטרף לקהילה שלנו, אנא בקרו באתר המגזין:

www.DigitalWhisper.co.il

"Talkin' bout a revolution sounds like a whisper"

הגליון הבא ייצא ביום האחרון של חודש יולי.

אפיק קסטיאל,

ניר אדר,

31.5.2016