

פיתוח מאחורי הקלעים

בקובץ הזה אני הולך לספר, לתאר ולהסביר את כל הלמידה וההתמודדות שלי בזמן פיתוח ה-CTF. ל-CTF שלי יש 5 שלבים (לא כולל שלב 0) ועבור כל שלב נתתי פרוט על התהליך.

שלב 0

המטרה:

יצירת נקודת הפתיחה שהוא מכתב עבור הסוכן החשאי (סוכן 07 – השם הזה היה נשמע לי מגניב).

הביצוע:

כתבתי את מכתב נקודת הפתיחה הפונה לסוכן (לא אשקר – נעזרתי ב-ChatGPT) ובסופו הוספתי קישור ל-הסנפה שיצרתי בשלב הבאה.



שלב 1

המטרה:

יצירת הסנפת Wireshark המכילה מספר חבילות SMTP שאחת מבניהם היא החבילה שמעניינת אותנו מפני שהיא מכילה את המסר המוצפן של שלב הבא. בנוסף, ההסנפה תכיל חבילה של HTTP עם פעולת GET שמבקשת קובץ תמונה בשם "XorKey" שתכיל מסתפר רנדומלי בגודל 16 ביטים (בחרתי 32767) שימשמש אותנו לשלב הבא (פיענוח המסר המוצפן).

הביצוע:

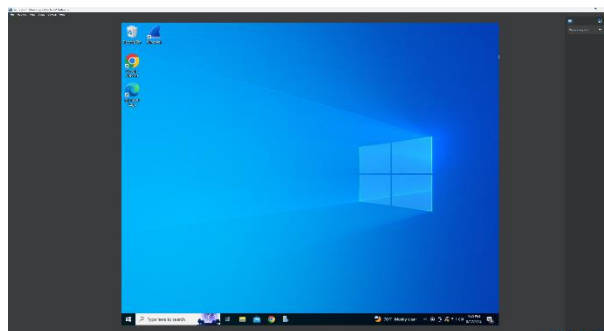
אם נבצע הסנפה בזמן שליחת מיילים עם Gmail, לא נקבל הודאות SMTP בכלל, וזאת מפני שבימינו כולן מוצפנות. אחרי חיפושים כושלים רבים אחר שרתי SMTP ישנים שלא משתמשים ב-TLS או שירותי SMTP מעל HTTP מצאתי פתרון אחר וטוב יותר, שאפילו ייתן לי גמישות רבה כמו, היכולת ליצור user-ים על שמות לבחירת, יצירת הודעות עם תוכן לבחירת ואפילו לבחור את דומיין השרת. הפתרון הוא בניית שרת SMTP מקומי (במחשב שלי) על דומיין שאני בוחר שיאפשר לחיבורים של מכשירים הנמצא באותו רשת LAN. אני אסביר ואתאר את התהליך.

הורדתי **VirtualBox** ו-**Windows 10** בקובץ IOS.

| | | | |
|-------------|--------------------|-----------------|--------------|
| Windows.iso | 16-Aug-24 11:26 AM | Disc Image File | 4,779,200 KB |
|-------------|--------------------|-----------------|--------------|



התקנתי את ה-Windows 10 על ה-VirtualBox וכך יצרתי מכונה וירטואלי על המחשב שהולך להכיל את השרת SMTP.

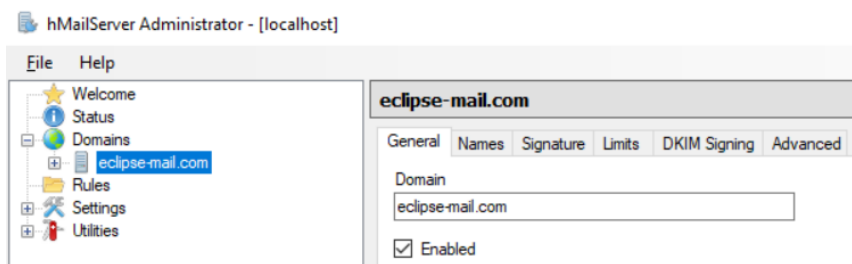
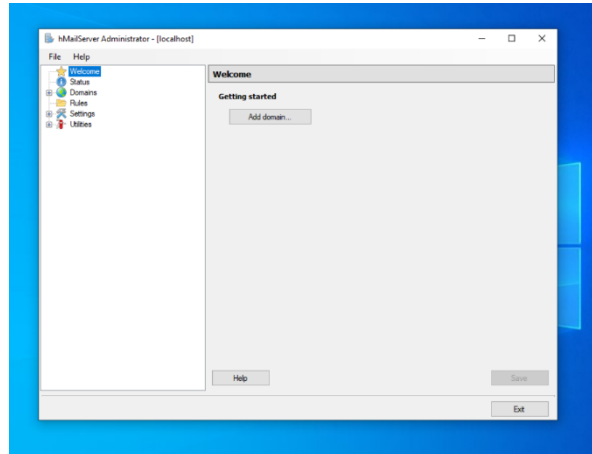


על המכונה הווירטואלית הורדתי את הדברים הבאים,

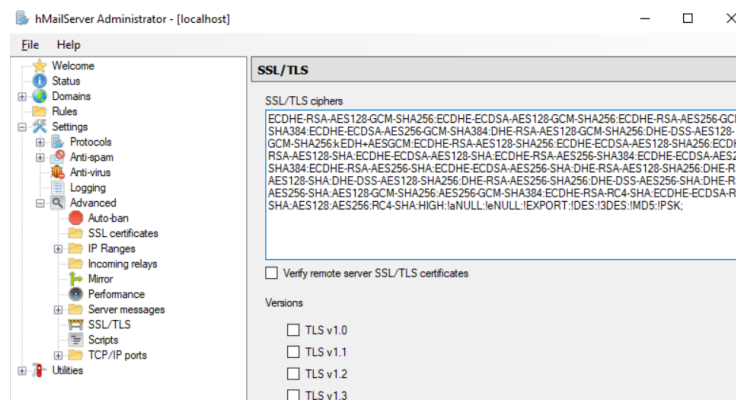
Google Chrome – כי לא היה.

hMailServer – שהיא תוכנה ליצירת שרתי מיילים.

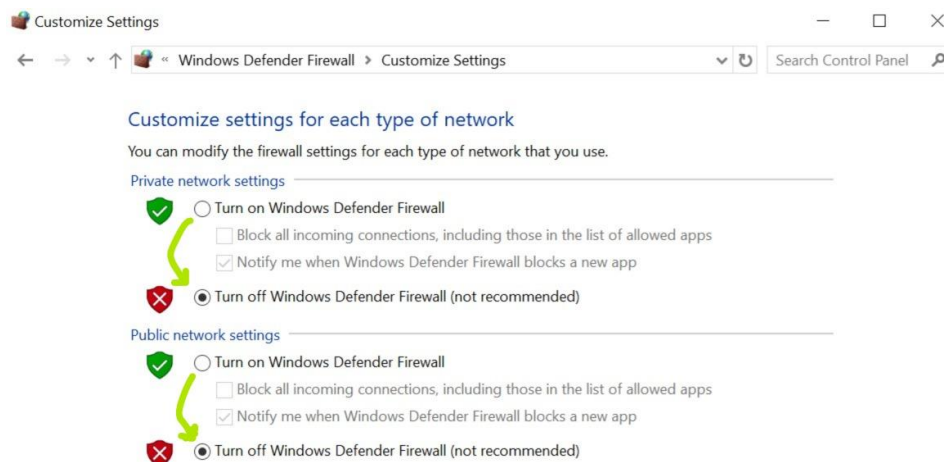
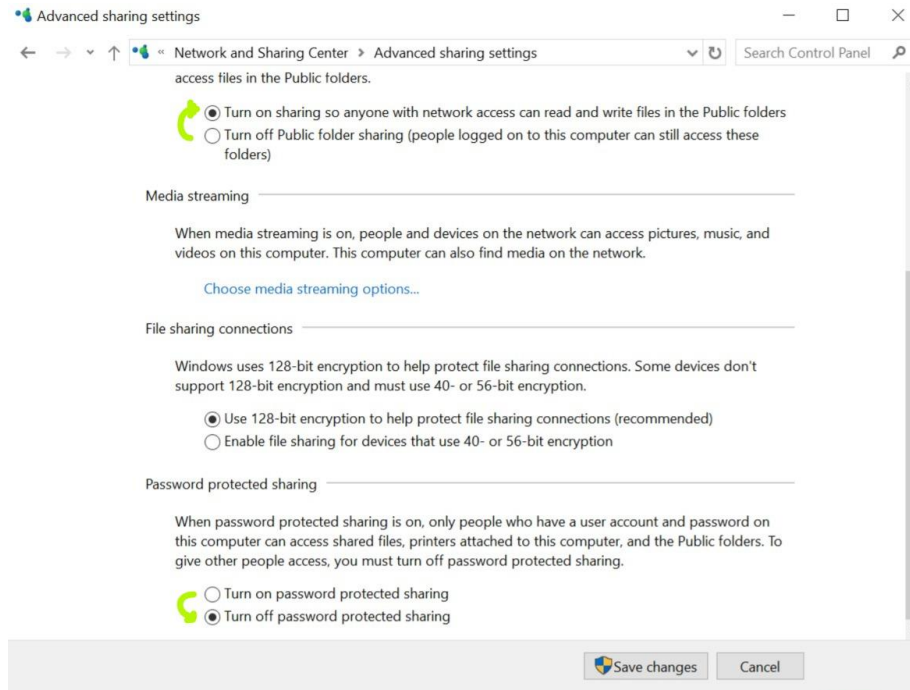
לאחר ההורדות הרצתי את hMailServer ויצרתי את השרת (לקח ממש כמה שניות בודדות) והוספתי דומיין לשרת. בחרתי בדומיין eclipse-mail.com,



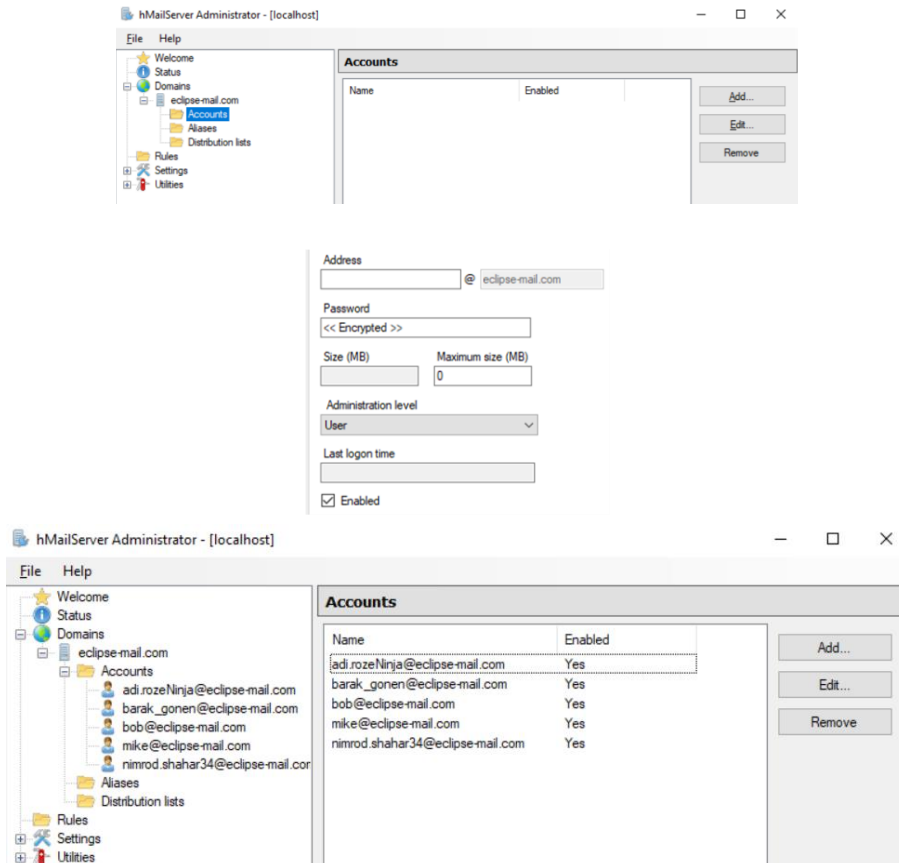
הגדרתי את כל ההגדרות של השרת (לא צריך לפרט), וכמובן החשוב מכל, דאגתי שהשרת לא ישתמש ב-TLS בכלל,



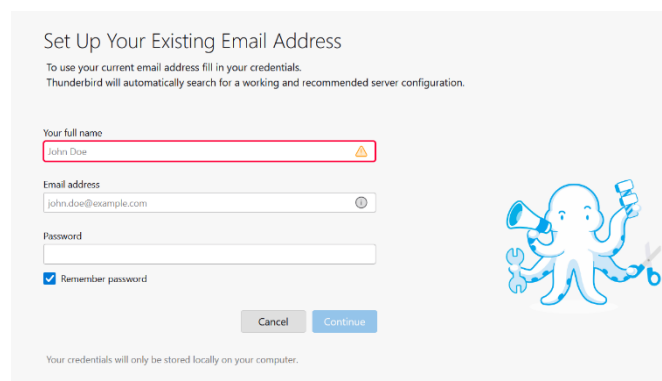
כדי לאפשר ללקוחות להתחבר אל השרת המכונה הווירטואלית צריכה לאפשר חיבורים אליה בעזרת כתובת ה- IP שלה (כמובן מדובר בחיבורים למכשירים באותו LAN). הלכתי להגדרות ואפשרתי זאת. בנוסף בטלתי את ה- firewall כדי שלא יחסמו הלקוחות בניסיונם להתחבר.



כעת השרת מוכן לפעולה, והדבר הבא הוא הוספת המשתמשים שיתקשרו בניהם בזמן ההסנפה. מבין כל המשתמשים הוספתי גם את 'bob' ואת 'mike' החשובים ליצירת הרמז לשלב הבא. בנוסף עבור כל משתמש יצרתי סיסמא,



לאחר הוספת כל המשתמשים צד השרת מוכן לגמרי. כעת צריך לחבר את כל המשתמשים לשרת בעזרת תוכנה בשם **Thunderbird** שהינה צעד הלקוח. בעזרתה אוכל לשלוח את ההודעות בזמן ההסנפה ולהחליט מה התוכן שלהן.



הורדתי את Thunderbird על כל המחשבים בבית (5 מחשבים) ולכל מחשב חיברתי משתמש אחד מתוך חמשת המשתמשים שכבר הגדרתי בשרת.

כדי לחבר את ה-Thunderbird אל השרת SMTP שיצרתי המחשב צריך לקשר את הדומיין של השרת אל כתובת ה-IP שלו, לכן הוספתי את ה-IP של המכונה הווירטואלית שבה נמצא השרת (שמצאתי בעזרת הרצת הפקודה ipconfig במכונה הווירטואלית) ואת הדומיין של השרת לתוך הקובץ hosts שנמצאת בנתיב הבא,

C:\Windows\System32\drivers\etc

```
hosts 16-Aug-24 6:52 PM File 2 KB

hosts - Notepad
File Edit Format View Help
# Copyright (c) 1993-2009 Microsoft Corp.
#
# This is a sample HOSTS file used by Microsoft TCP/IP for Windows.
#
# This file contains the mappings of IP addresses to host names. Each
# entry should be kept on an individual line. The IP address should
# be placed in the first column followed by the corresponding host name.
# The IP address and the host name should be separated by at least one
# space.
#
# Additionally, comments (such as these) may be inserted on individual
# lines or following the machine name denoted by a '#' symbol.
#
# For example:
#
#       102.54.94.97       rhino.acme.com       # source server
#       38.25.63.10      x.acme.com           # x client host

# localhost name resolution is handled within DNS itself.
#       127.0.0.1         localhost
#       ::1              localhost

172.30.1.154 mail.eclipse-mail.com
172.30.1.154 eclipse-mail.com
172.30.1.154 smtp.mail.eclipse-mail.com
172.30.1.154 pop3.mail.eclipse-mail.com
```

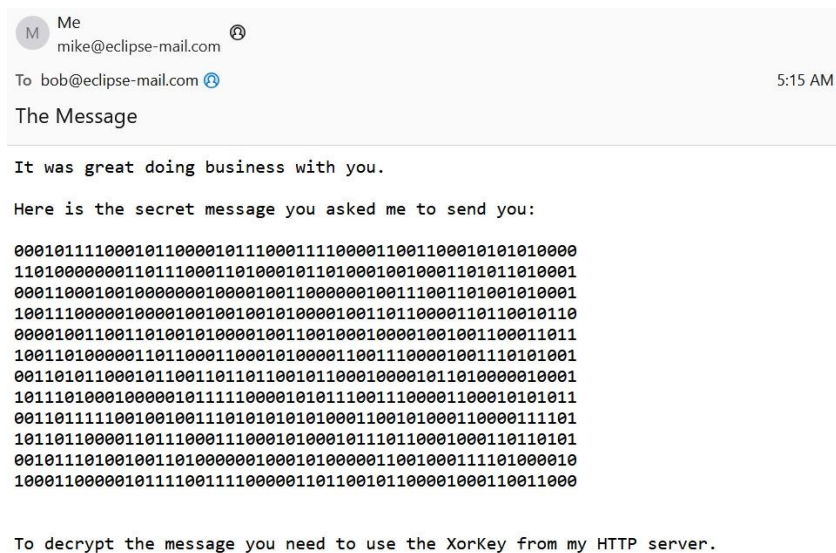
פעולה זאת תאפשר למחשב להתחבר אל הדומיין של השרת בעזרת כתובת ה-IP שלו (DNS מקומי).

כעת כל מחשב מחובר למשתמש וכולם מוכנים לשלוח ולקבל הודעות. עבור ההסנפה יצרתי עלילה והכנתי את כל ההודעות לשליחה מראש,

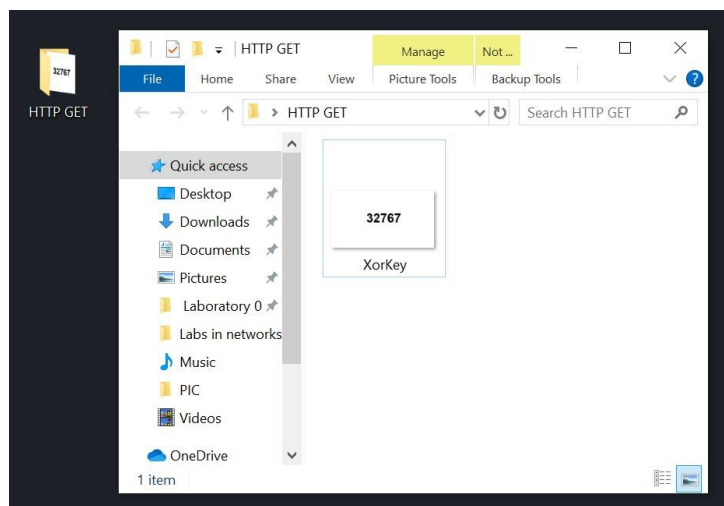
Me
barak_gonen@eclipse-mail.com
To nimrod.shahar36@eclipse-mail.com, mike@eclipse-mail.com MORE 5:15 AM
Summary Exercise
Hi students,
Before you start the final exercise I just wanted to start with success blessing.
I had a great semester and really enjoyed teaching you.
I wish you the best of luck for the exercise and for your future life.
good luck!!!

Me
mike@eclipse-mail.com
To nimrod.shahar36@eclipse-mail.com 5:15 AM
I'm So Sorry!
What's up Nimrod?
I just wanted to tell you that I'm very sorry bro.
I helped bob the hacker steal some file because I owed money.
I hope you will forgive me!
goodbye.

ישנם עוד כמה תמונות, אך התמונה החשובה ביותר היא ההודעה ש- mike שולח ל- bob המכילה את המסר המוצפן,



כעת כל ההודעות מוכנות לשליחה. ומה שנישאר זה להכין את התמונה שתכיל את המפתח לפיענוח המסר המוצפן שנמצא בהודעה ש- mike שולח ל- bob. פתחתי Word וכתבתי את מספר המפתח הרנדומלי שבחרתי (המספר 32767) ולקחתי תמונת מסך של המספר ושמרתי את התמונה בשם XorKey.png בתקיה בשם HTTP GET.



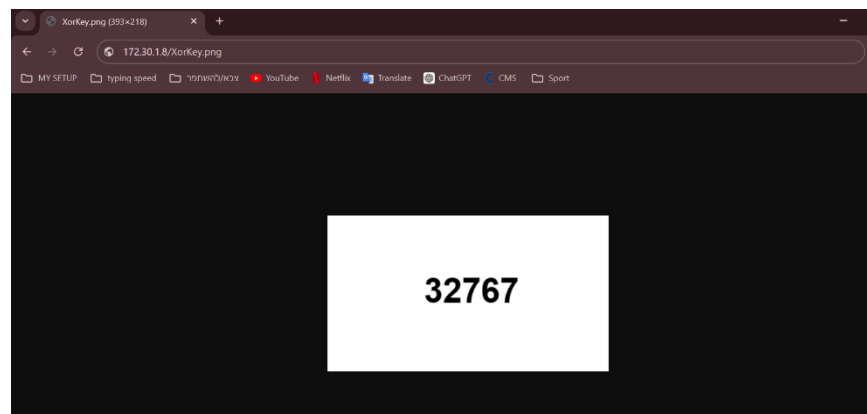
כעת עלי היה למצוא דרך לבצע בקשת GET של התמונה בזמן הסנפה. הפתרון היה להשתמש במחשב הנייד שלי כ- HTTP Server והמחשב הנייד שלי כ- Client ולבצע בקשת GET עבור התמונה. לכן נכנסתי להגדרות של המחשב הנייד שלי, צד השרת, ואפשרתי התחברות אליו בעזרת כתובת ה- IP שלו ובנוסף ביטלתי את ה- firewalls. פעולות אלה כבר מוכרות לנו, מפני שעשינו אותן עבור המכונה הווירטואלית שמכילה את השרת SMTP.

לאחר שהמחשב הנייד, (צד השרת) מוכן לקבל חיבורים, הרצתי ב- CMD את הפקודה הבאה במיקום בו מיקמנו את התמונה,

python -m http.server 80

פקודה זאת גורמת למחשב ליצור שרת HTTP בפורט 80 בתקיה בו הוא נימצא. כעת צד השרת מוכן. דרך המחשב הנייד (צד הלקוח) אני נכנס לדפדפן, וכותב את ה- IP של השרת (המחשב הנייד), את הפורט 80 בו הוא מאזין ואת התמונה,

http://172.30.1.8:80/XorKey.png



פעולה זאת מבצעת בקשת GET ומקבל את התמונה.

```
C:\WINDOWS\system32\cmd.exe - python -m http.server 80

C:\Users\eliel\Desktop\HTTP GET>python -m http.server 80
Serving HTTP on :: port 80 (http://[::]:80/) ...
::ffff:172.30.1.220 - - [18/Aug/2024 05:37:16] "GET /XorKey.png HTTP/1.1" 304 -
```


כעת הכל מוכן להסנפה. פתחתי Wireshark במחשב הנייח שלי והתחלתי הסנפה. בזמן ההסנפה שלחתי את כל ההודעות שהכנתי מראש ובצעתי את בקשת התמונה שגם אותה הכנתי מראש. עצרתי את ההסנפה בסוף התהליך ובדקתי שכל החבילות נמצאות, וכך היה. עד כה סיימתי את יצירת ההסנפה ואת השלב הראשון.

שלב 2

המטרה:

בעזרת המסר המוצפן והמפתח שיצרתי בשלב קודם ניצור קוד python, שהינו גם פתרון השלב, שלוקח את המפתח ואת המסר המוצפן ומפענח את המסר בעזרת הצפנה סימטרית בשיטת XOR, כפי שכבר עשינו בתרגיל בית 6.

הביצוע:

לקחתי את הקוד שכתבתי בתרגיל 6 (הצפנה סימטרית בעזרת XOR) והתאמתי אותו למקרה שלי.

```
def symmetric_encryption(input_data, key):  
    # Determine chunk and key size based on length of data (even or odd)  
    if len(input_data) % 2 == 0:  
        chunk_size = 16  
        # Using the all bits of key  
        key_in_bits = format(key, '016b')  
    else:  
        chunk_size = 8  
        # Using the last 8 bits of the key  
        key_in_bits = format(key, '016b')[-8:]  
  
    # Split data into chunks of appropriate size  
    chunks = [input_data[i:i + chunk_size] for i in range(0, len(input_data), chunk_size)]  
  
    # Perform XOR operation on each chunk with the key  
    result = ""  
    for chunk in chunks:  
        for bit, key_bit in zip(chunk, key_in_bits):  
            result += '1' if bit != key_bit else '0'  
  
    return result
```

```

Encrypted_message =
'000101111000101100001011100011110000110011000101010100001101000000011
0111000110100010110100010010001101011010001000110001001000000010000100
1100000010011100110100101000110011100000100001001001001010000100110110
0001101100101100000100110011010010100001001100100010000100100110001101
1100110100000110110001100010100001100111000010011101010010011010110001
0110011011011001011000100001011010000010001101110100010000010111110000
1010111001110000110001010101100110111110010010011101010101010001100101
0001100001111011011011000011011100011100010100010111011000100011011010
1001011101001001101000000100010100000110010001111010000101000110000010
1111001111000001101100101100001000110011000'

XorKey = 32767

Decoded_message = symmetric_encryption(Encrypted_message, XorKey)

message = ''.join(chr(int(Decoded_message[i:i + 8], 2)) for i in
range(0, len(Decoded_message), 8))

print(message)

```

בעזרת הקוד הזה, נקבל את קישור הדרייב שמכיל את השלב הבא. כך סיימתי בהצלחה ובקלות את
שלב 2 (היה יחסית קל – לא?).

שלב 3

המטרה:

יצירת שרת HTTP מקומי בקובץ exe המאזין לפורט שאינו ידוע (אבל לי הוא כן – אהאה). יצירת קוד לקוח מתאים לשרת (זה גם התפקיד של פותר ה-CTF), ומימוש טוב של קוד הלקוח ייתן קישור לאתר שהיינו השלב הבא.

הביצוע:

חזרתי ולמדתי על המבנה של בקשת GET בפרוטוקול HTTP ויצרתי בקשה משלי,

GET bob/website/link HTTP/1.0\r\nHost: bob.com\r\n\r\n

על פי הבקשה הזאת, יצרתי שרת HTTP שעבור הודעה כזאת הוא יחזיר קוד **200 ok** ותוכן הודעה יהיה קישור לאתר (שאבנה בשלב הבא). האתר יהיה בעצם המערכת אחסון של bob המאוחסן את הקובץ הגנוב.

כמובן, פותר ה-CTF לא יודע את גרסת ה-HTTP בו משתמש השרת, או את הצורך בשדה ה-Host ולכן יצרתי הודעות שגיאה עבור מקרים שונים של שליחת הודעה שגויה, לדוגמא, שדה או ערך שדה שגוי או פעולה לא נתמכת וכדומה. כמובן שפותר ה-CTF לא יודע על קיומו של הנתיב 'bob/website/link' ועליו לנסות לבצע בקשת GET **לעמוד ברת המחדל** ('') ועבור בקשה זו הוא יקבל קוד 302 שיפנה אותו למיקום 'bob/website/link'.

קוד השרת:

```
import socket

IP = '127.0.0.1'
PORT = 46285

def handle_client(client_socket):
    data = client_socket.recv(1024).decode()

    if not data:
        client_socket.send("400 Bad Request Empty Request\r\nContent-
Length: 0\r\n\r\n".encode())
        return
```

```

arguments = data.split()
try:
    if arguments[0] != 'GET':
        client_socket.send("405 Method Not Allowed\r\nContent-
Length: 0\r\n\r\n".encode())
        return

except Exception:
    client_socket.send("400 Bad Request Method Missing\r\nContent-
Length: 0\r\n\r\n".encode())
    return

try:
    version = arguments[2].split('/')
    if version[0] != 'HTTP' or version[1] != '1.0':
        client_socket.send("505 HTTP Version Not
Supported\r\nContent-Length: 0\r\n\r\n".encode())
        return

except Exception:
    client_socket.send("400 Bad Request HTTP Version
Missing\r\nContent-Length: 0\r\n\r\n".encode())
    return

try:
    if arguments[3] != 'Host:':
        client_socket.send("400 Bad Request Host Field
Require\r\nContent-Length: 0\r\n\r\n".encode())
        return

except Exception:
    client_socket.send("400 Bad Request Host Missing\r\nContent-
Length: 0\r\n\r\n".encode())
    return

try:
    if arguments[4] != 'bob.com':
        client_socket.send("400 Bad Request Host Field
Invalid\r\nContent-Length: 0\r\n\r\n".encode())
        return

except Exception:
    client_socket.send("400 Bad Request Host Field
Missing\r\nContent-Length: 0\r\n\r\n".encode())
    return

try:
    if arguments[1] == '/':
        client_socket.send("HTTP/1.0 302 Found\r\nLocation:
bob/website/link\r\n\r\n".encode())
        return

```

```

        if arguments[1] == 'bob/website/link':
            client_socket.send("HTTP/1.0 200 OK\r\nContent-Length:
0\r\n\r\nhttps://bobbase.vercel.app".encode())
            return

    except Exception:
        client_socket.send("400 Bad Request Requested Item
Missing\r\nContent-Length: 0\r\n\r\n\r\n".encode())
        return

    client_socket.send("404 Not Found\r\nContent-Length:
0\r\n\r\n\r\n".encode())

def main():
    server_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    server_socket.bind((IP, PORT))
    server_socket.listen(1)
    print(f"Server \'bob.com\' listening to connection on localhost")

    while True:
        client_socket, client_address = server_socket.accept()
        print(f"Connection success from {client_address}")
        handle_client(client_socket)

if __name__ == '__main__':
    main()

```

את קוד השרת הפכתי לקובץ הרצה exe בעזרת הרצת הפקודה הבאה:

pyinstaller server.py - - onefile

את קובץ ההרצה של השרת הכנסתי ל- Google Drive שאת הקישור כבר יצרתי בשלב 1.

כמובן שאחרי פיתוח השרת יש לפתח את קוד הלקוח. פיתוח קוד הלקוח היינו קצר ופשוט אך דורש שכתובת ההודעה הנשלחת תעשה כמו שצריך. בנוסף לכך ניתן לראות שהפורט של השרת לא נמסר לפותר ה- CTF ולכן על הפותר למצוא אותו בעזרת שימוש בפקודה Nmap בזמן הרצת השרת. פעולה זו תיתן לפותר את הפורט הפתוח שהשרת משתמש בו.

```
C:\Users\MSI>nmap -p- localhost
Starting Nmap 7.95 ( https://nmap.org ) at 2024-08-25 23:54 Jerusalem Daylight Time
Nmap scan report for localhost (127.0.0.1)
Host is up (0.000056s latency).
Other addresses for localhost (not scanned): ::1
rDNS record for 127.0.0.1: kubernetes.docker.internal
Not shown: 65510 closed tcp ports (reset)
PORT      STATE      SERVICE
135/tcp    open       msrpc
137/tcp    filtered  netbios-ns
445/tcp    open       microsoft-ds
1536/tcp   open       ampr-inter
1537/tcp   open       sdsc-lm
1538/tcp   open       3ds-lm
1539/tcp   open       intellistor-lm
1540/tcp   open       rds
1543/tcp   open       simba-cs
2179/tcp   open       vmrdp
5040/tcp   open       unknown
5357/tcp   open       wsdaapi
5939/tcp   open       unknown
6463/tcp   open       unknown
7680/tcp   open       pando-pub
9010/tcp   open       sdr
9080/tcp   open       glrpc
9100/tcp   open       jetdirect
9180/tcp   open       unknown
15924/tcp  open       unknown
45600/tcp  open       unknown
45654/tcp  open       unknown
46285/tcp  open       unknown
63342/tcp  open       unknown
65333/tcp  open       unknown

Nmap done: 1 IP address (1 host up) scanned in 2.71 seconds
```

הינה קוד הלקוח שפתחתי המתאים לשרת,

```
import socket

IP = '127.0.0.1'
PORT = 46285

def main():
    client_socket = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
    client_socket.connect((IP, PORT))

    request = (
        "GET bob/website/link HTTP/1.0\r\n"
        "Host: bob.com\r\n"
        "\r\n"
    )
    client_socket.send(request.encode())

    response = client_socket.recv(1024).decode()
    print("Server response:")
    print(response)

    client_socket.close()
```

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

קוד הלקוח ישמש בנוסף כחלק מהפתרון ל- CTF.

כעת סיימנו את השלב ועלינו לגשת לשלב הבא, בניית האתר.

שלב 4

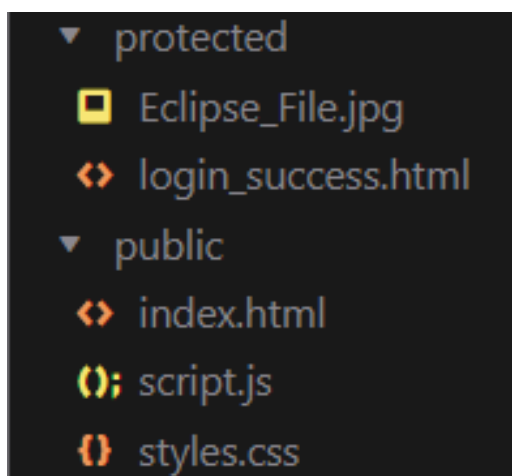
המטרה:

יצירת אתר. האתר מכיל Login page, Login success page ומספר נתיבים מוסתרים. אחד הנתיבים יכיל הסנפת Wireshark ונתיב אחר יכיל Random Client ומפתח סימטרי לפיענוח ההסנפה שאצור בשלב הבא. בשלב זה אני יוצר את האתר והנתיבים ומריץ את הפקודה Gobaster על האתר למציאת הנתיבים המוסתרים ושני הנתיבים מתוכם החשובים לשלב הבא.

הביצוע:

הערה – הוספתי את קוד האתר בקובץ zip.

עבור השלב הזה הייתי צריך לחדד את כישרונותי בתכנות אתרים. יצרתי Login page שמכיל שני תבניות לקליטת השם משתמש והסיסמא, ויצרתי Login success page שהוא העמוד שיופיע לאחר פעולת ההתחברות ויכיל כפתור להורדת התמונה ומסר הצלחה ב- CTF.



```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Login Page</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>

  <canvas id="matrix"></canvas>

  <div class="login-container">
    <h1>Access Denied</h1>
    <form id="loginForm">
      <input type="text" name="username" placeholder="Username"
required><br>
      <input type="password" name="password" placeholder="Password"
required><br>
      <input type="submit" value="Login">
    </form>
    <p id="error-message" style="color: red;"></p>
  </div>
  <script src="script.js"></script>
</body>
</html>
```

```
body {
  margin: 0;
  padding: 0;
  font-family: 'Courier New', Courier, monospace;
  background-color: #000;
  color: #0f0;
  display: flex;
  justify-content: center;
  align-items: center;
  height: 100vh;
  overflow: hidden;
  position: relative;
}

canvas {
  position: absolute;
  top: 0;
  left: 0;
  z-index: -1;
}

.Login-container {
  border: 2px solid #0f0;
  padding: 20px;
  width: 320px;
  background-color: rgba(0, 0, 0, 0.8);
  box-shadow: 0 0 20px #0f0;
  position: relative;
  animation: flicker 1.5s infinite alternate;
}

@keyframes flicker {
  0% { opacity: 1; }
  50% { opacity: 0.9; }
  100% { opacity: 1; }
}

.Login-container h1 {
  text-align: center;
  margin-bottom: 20px;
  font-size: 28px;
  position: relative;
  color: #ff0;
}
```

```

text-shadow: 0 0 10px #f00, 0 0 20px #0f0;
animation: glitch 1.5s infinite;
}

@keyframes glitch {
  0% {
    text-shadow: 2px 2px 0 #0f0, -2px -2px 0 #0f0;
  }
  25% {
    text-shadow: 2px -2px 0 #f00, -2px 2px 0 #0f0;
  }
  50% {
    text-shadow: -2px -2px 0 #00f, 2px 2px 0 #f00;
  }
  75% {
    text-shadow: -2px 2px 0 #f00, 2px -2px 0 #00f;
  }
  100% {
    text-shadow: 2px 2px 0 #0f0, -2px -2px 0 #0f0;
  }
}

input[type="text"], input[type="password"] {
  width: calc(100% - 24px); /* Adjusted width */
  padding: 12px;
  margin: 10px 0;
  background-color: #111;
  color: #0f0;
  border: 1px solid #0f0;
  box-shadow: 0 0 10px #0f0;
  font-size: 16px;
}

input[type="text"]::placeholder,
input[type="password"]::placeholder {
  color: #666;
}

input[type="submit"] {
  width: 100%;
  padding: 12px;
  background-color: #0f0;
  color: #000;
  border: none;
  font-weight: bold;
}

```

```

    cursor: pointer;
    transition: background-color 0.3s ease;
    box-shadow: 0 0 10px #0f0;
    font-size: 16px;
}

input[type="submit"]:hover {
    background-color: #0a0;
}

```

קובץ ה-script.js

```

const canvas = document.getElementById('matrix');
const ctx = canvas.getContext('2d');
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;

const letters =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789@#%^&*()";
const fontSize = 16;
const columns = canvas.width / fontSize;

const drops = Array(Math.floor(columns)).fill(1);

function draw() {
    ctx.fillStyle = "rgba(0, 0, 0, 0.05)";
    ctx.fillRect(0, 0, canvas.width, canvas.height);
    ctx.fillStyle = "#0f0";
    ctx.font = fontSize + "px monospace";

    for (let i = 0; i < drops.length; i++) {
        const text = letters.charAt(Math.floor(Math.random() * letters.length));
        ctx.fillText(text, i * fontSize, drops[i] * fontSize);

        if (drops[i] * fontSize > canvas.height && Math.random() > 0.975) {
            drops[i] = 0;
        }

        drops[i]++;
    }
}

setInterval(draw, 33);

```

```
<!DOCTYPE html>
<html Lang="he">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Operation Success</title>
  <style>
    body {
      background-color: #000;
      color: #0f0;
      font-family: 'Courier New', Courier, monospace;
      display: flex;
      justify-content: center;
      align-items: center;
      height: 100vh;
      margin: 0;
      text-align: center;
      overflow: hidden;
      position: relative;
    }
    .container {
      position: relative;
      padding: 20px;
      border: 3px solid #0f0;
      box-shadow: 0 0 20px #0f0;
      z-index: 2;
      border-radius: 15px;
      background: rgba(0, 0, 0, 0.7);
      margin-top: 150px;
    }
    h1 {
      font-size: 6rem;
      margin: 0;
      position: relative;
      z-index: 3;
      animation: glitch 1.5s infinite alternate;
    }
    p {
      font-size: 1.2rem;
      margin-top: 20px;
      line-height: 1.6;
      letter-spacing: 1px;
      white-space: pre-wrap;
    }
  </style>
</head>
<body>
  <div class="container">
    <h1>Operation Success</h1>
    <p>...</p>
  </div>
</body>
</html>
```

```
    position: relative;
    z-index: 3;
    animation: scramble 4s infinite alternate;
}

.download-link {
    display: inline-block;
    margin-top: 30px;
    padding: 10px 20px;
    border: 2px solid #0f0;
    color: #0f0;
    text-decoration: none;
    font-size: 1.1rem;
    transition: background-color 0.3s, color 0.3s;
    background: #000;
    box-shadow: 0 0 10px #0f0, inset 0 0 5px #0f0;
    animation: pulse 1s infinite alternate;
}

.download-link:hover {
    background-color: #0f0;
    color: #000;
}

@keyframes glitch {
    0% { transform: skew(-5deg); }
    20% { transform: skew(5deg); }
    40% { transform: translate(-5px, -5px); }
    60% { transform: translate(5px, 5px); }
    80% { transform: translate(0, 0); }
    100% { transform: skew(0deg); }
}

@keyframes scramble {
    0% { transform: translateY(0px); opacity: 1; }
    50% { transform: translateY(-10px); opacity: 0.7; }
    100% { transform: translateY(0px); opacity: 1; }
}

@keyframes pulse {
    0%, 100% { box-shadow: 0 0 10px #0f0, inset 0 0 5px #0f0; }
    50% { box-shadow: 0 0 20px #0f0, inset 0 0 10px #0f0; }
}

.background {
    position: absolute;
    top: 0;
```

```

    left: 0;
    width: 100%;
    height: 100%;
    background: radial-gradient(circle, rgba(0, 255, 0, 0.1), rgba(255,
0, 0, 0.1));
    background-size: 200% 200%;
    animation: backgroundShift 5s infinite linear;
    z-index: 1;
}

@keyframes backgroundShift {
    0% { background-position: 0 0; }
    50% { background-position: 100% 100%; }
    100% { background-position: 0 0; }
}

.hacker-hood {
    position: absolute;
    top: 15%;
    left: 50%;
    width: 250px;
    height: 250px;
    background-color: #111;
    border-radius: 0 0 125px 125px;
    box-shadow: 0 0 20px rgba(0, 255, 0, 0.7);
    transform: translateX(-50%);
    z-index: 0;
    clip-path: polygon(0 0, 100% 0, 50% 100%);
}

.hacker-hood::before {
    content: '';
    position: absolute;
    top: 10%;
    left: 50%;
    width: 60px;
    height: 60px;
    background-color: #0f0;
    border-radius: 50%;
    transform: translateX(-50%);
    box-shadow: 0 0 15px rgba(0, 255, 0, 0.8);
}

.hacker-hood::after {
    content: '';
    position: absolute;
    top: 70%;

```



```

    left: 50%;
    width: 180px;
    height: 180px;
    background-color: rgba(0, 255, 0, 0.2);
    border-radius: 50%;
    transform: translateX(-50%);
    box-shadow: 0 0 30px rgba(0, 255, 0, 0.5);
  }
</style>
</head>
<body>
  <div class="background"></div>
  <div class="hacker-hood"></div>
  <div class="container">
    <h1>Well Done<br>Operation Success</h1>
    <p>ובינתיים, תוכל לקבל <br>חשבת שזה נגמר? הבלאגן רק מתחיל<br>ברכותיי, אתה חכם. אבל לא חכם כמוני<p>
    <p>נתראה בקרוב<br>בחזרה את הקובץ שגנבתי<br>
    <a href="Eclipse_File.jpg" download="Eclipse_File.jpg" class="download-link">Eclipse File</a>
  </div>
</body>
</html>

```

התמונה (יצרתי בעזרת AI):



עכשיו צריך להוסיף את פעולת התחברות לאתר על ידי הכנסת שם משתמש וסיסמא. יצרתי שם משתמש וסיסמא:

שם משתמש ← **BobAdmin**

סיסמא ← **Y0uG0tMe#Unlock!**

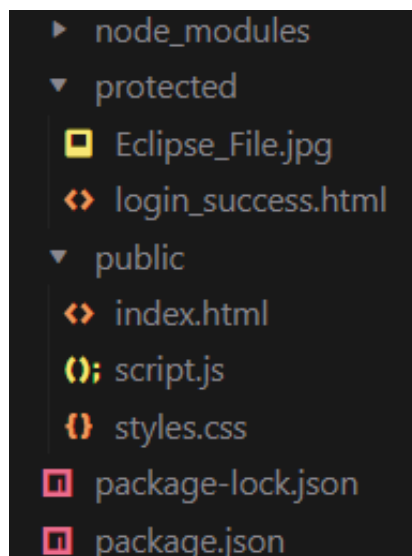
לאחר יצירת שם המשתמש והסיסמא נתקלתי בבעיה. היכן אשמור את השם המשתמש והסיסמא? אם אני אשמור אותם בקוד ה- `script.js` שכתבתי, כל גולש, בעזרת `Inspect (F12)` יוכל לראות את תוכן הקוד ולמצוא את השם משתמש והסיסמא ולהתחבר, דבר שאני רוצה למנוע. לאחר חקירה ארוכה מאוד מצאתי פתרון לבעיה. השתמשתי ב- **Node.js** ליצירת קוד שרת שתפקידו יהיה לשמור את השם המשתמש והסיסמא ולבצע בדיקת שם המשתמש והסיסמא בזמן ההתחברות.

הורדתי את `Node.js` והתקנתי אותו על המחשב. לאחר ההתקנה, פתחתי `cmd` במיקום קבצי האתר שיצרתי והרצתי את הפקודות הבאות:

```
npm init -y
```

```
npm install express body-parser
```

פקודות אלו הורידו את החבילה **express body-parser** של `Node.js` לאזור קבצי האתר שיצרתי.



צעד הבא הוא יצירת השרת. יצרתי קובץ חדש וקראתי לו **"server.js"** ובו שמרתי את שם המשתמש והסיסמא ובצעתי את בדיקה שם המשתמש והסיסמא בזמן התחברות. להלן קוד השרת ועדכון קוד הקובץ `script.js` ששולח לשרת את שם המשתמש והסיסמא שנקלטו.

```
const express = require('express');
const bodyParser = require('body-parser');
const path = require('path');
const app = express();

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.use(express.static(path.join(__dirname, 'public')));

const validUsername = 'BobAdmin';
const validPassword = 'Y0uG0tMe#Unlock!';

app.post('/login', (req, res) => {
  const { username, password } = req.body;

  if (username === validUsername && password === validPassword) {
    res.send({ success: true });
  } else {
    res.send({ success: false });
  }
});

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

app.listen(3000, () => {
  console.log('Server is running on http://localhost:3000');
});
```

קוד ה- script.js (המעודכן):

```
const canvas = document.getElementById('matrix');
const ctx = canvas.getContext('2d');
canvas.width = window.innerWidth;
canvas.height = window.innerHeight;

const letters =
"ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789@#%^&*()";
const fontSize = 16;
const columns = canvas.width / fontSize;

const drops = Array(Math.floor(columns)).fill(1);

function draw() {
  ctx.fillStyle = "rgba(0, 0, 0, 0.05)";
  ctx.fillRect(0, 0, canvas.width, canvas.height);
  ctx.fillStyle = "#0f0";
  ctx.font = fontSize + "px monospace";

  for (let i = 0; i < drops.length; i++) {
    const text = letters.charAt(Math.floor(Math.random() * letters.length));
    ctx.fillText(text, i * fontSize, drops[i] * fontSize);

    if (drops[i] * fontSize > canvas.height && Math.random() > 0.975) {
      drops[i] = 0;
    }

    drops[i]++;
  }
}

setInterval(draw, 33);

document.getElementById('loginForm').addEventListener('submit', function(event) {
  event.preventDefault(); // Prevent form submission

  const username = event.target.username.value;
  const password = event.target.password.value;

  fetch('/login', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
  },
```

```

        body: JSON.stringify({ username, password })
    })
    .then(response => response.json())
    .then(data => {
        if (data.success) {
            window.location.href = '/protected/login_success.html';
        } else {
            document.getElementById('error-message').textContent = 'Invalid
username or password!';
        }
    });
});
});

```

כעת פעולת ההתחברות לאתר פועלת כמו שצריך! אומנם כך חשבת. זאת מפני ששמתי לב לבעיה. הבעיה הייתה שיכולתי לגשת לעמוד Login success page, שאמור להתקבל לאחר התחברות, בעזרת הכנסת URL מתאים מבלי לבצע התחברות, דבר שאני מאוד רוצה למנוע.

כדי לפתור את הבעיה עדכנתי את קוד השרת כך שכל בקשה למשאב הנמצא בקובץ protected לא תתקבל ללא התחברות. כדי שדבר זה יעבוד, הייתי צריך דאוג שהשרת יוכל לשמור את עצמו על מצב "מחובר" לאחר התחברות. כדי לבצע זאת הורדתי חבילת Node.js נוספת בשם **express-session** בעזרת הפקודה:

npm install express-session

כמוכן שעדכנתי את קוד השרת כפי שציינתי.

קוד ה- **server.js** (המעודכן):

```

const express = require('express');
const bodyParser = require('body-parser');
const session = require('express-session');
const path = require('path');
const app = express();

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));

app.use(session({
    secret: 'b2f8c8e9b6c7e0d4f3b2d6e8a7e0c4a5b9f3c8d7e2f9b3c7a5d8e9a0f2b6c7e8d',
    resave: false,
    saveUninitialized: true,
})));

```

```

app.use(express.static(path.join(__dirname, 'public')));

function ensureAuthenticated(req, res, next) {
  if (req.session.authenticated) {
    return next();
  }
  res.redirect('/');
}

const validUsername = 'BobAdmin';
const validPassword = 'Y0uG0tMe#Unlock!';

app.post('/login', (req, res) => {
  const { username, password } = req.body;

  if (username === validUsername && password === validPassword) {
    req.session.authenticated = true;
    res.send({ success: true });
  } else {
    res.send({ success: false });
  }
});

app.get('/protected/*', ensureAuthenticated, (req, res) => {
  const filePath = path.join(__dirname, 'protected', req.params[0]);
  res.sendFile(filePath);
});

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

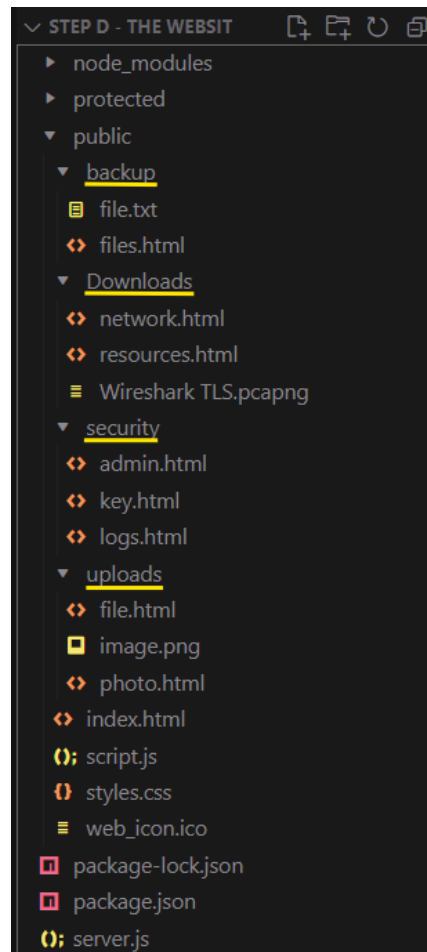
app.listen(3000, () => {
  console.log('Server is running on http://localhost:3000');
});

```

כעת נוכל להגיד סופית שפעולת ההתחברות לאתר פועלת כמו שצריך.

לאחר סיום בניית האתר הבסיסי עברתי ליצירת הנתיבים המוסתרים (שאינם ידועים לגולש) ושימוש ב-Gobaster tool כדי למצוא אותם. לכן ישבתי מספר שעות (לא מעט) ויצרתי מספר נתיבים (8 בסך הכל) ולכל נתיב יצרתי עמוד html. כמובן רק שניים מהנתיבים יכילו מידע חשוב, ההסנפה והמפתח לפיענוח ההסנפה. את שניהם ניצור בשלב הבא, והשאר לא יכילו דבר (dead ends).

לכל קובץ שיצרתי קראתי בשם (כך יצרתי את הנתביים). לא עליתי את קוד הנתבי לקובץ מפני שזה חשוב (וארוך מידי), אך אם תרצו לראות בכל זאת, הוא ימצא בקובץ נפרד, אך חשוב לראות את מבנה הקוד הסופי שיצרתי,



כמובן שעדכנתי את השרת עבור כל נתיב חדשים שיצרתי והוספתי את קוד השגיאה 404 Not Found.

קוד ה- server.js (המעודכן סופית):

```
const express = require('express');
const bodyParser = require('body-parser');
const session = require('express-session');
const path = require('path');
const app = express();

app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: true }));
app.use(session({
```

```
    secret: 'b2f8c8e9b6c7e0d4f3b2d6e8a7e0c4a5b9f3c8d7e2f9b3c7a5d8e9a0f2b6c7e8d',
    resave: false,
    saveUninitialized: true,
  }));

app.use(express.static(path.join(__dirname, 'public')));

function ensureAuthenticated(req, res, next) {
  if (req.session.authenticated) {
    return next();
  }
  res.redirect('/');
}

const validUsername = 'BobAdmin';
const validPassword = 'Y0uG0tMe#Unlock!';

app.post('/login', (req, res) => {
  const { username, password } = req.body;

  if (username === validUsername && password === validPassword) {
    req.session.authenticated = true;
    res.send({ success: true });
  } else {
    res.send({ success: false });
  }
});

app.get('/protected/*', ensureAuthenticated, (req, res) => {
  const filePath = path.join(__dirname, 'protected', req.params[0]);
  res.sendFile(filePath);
});

app.get('/backup/files', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'backup', 'files.html'));
});

app.get('/backup/file.txt', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'backup', 'file.txt'));
});

app.get('/Downloads/network', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'Downloads', 'network.html'));
});
```



```
app.get('/Downloads/secret_file.png', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'Downloads', 'secret_file.png'));
});

app.get('/Downloads/resources', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'Downloads', 'resources.html'));
});

app.get('/security/admin', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'security', 'admin.html'));
});

app.get('/security/key', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'security', 'key.html'));
});

app.get('/security/logs', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'security', 'logs.html'));
});

app.get('/uploads/file', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'uploads', 'file.html'));
});

app.get('/uploads/photo', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'uploads', 'photo.html'));
});

app.get('/uploads/image.png', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'uploads', 'image.png'));
});

app.get('/', (req, res) => {
  res.sendFile(path.join(__dirname, 'public', 'index.html'));
});

app.use((req, res) => {
  res.status(404).send('404 Not Found');
});

app.listen(3000, () => {
  console.log('Server is running on http://localhost:3000');
});
```

האתר כעת מוכן ועובד. הצעד הבא הוא להעלות את האתר לאינטרנט כך שכל אחד יוכל לגשת אליו. כדי לבצע זאת השתמשתי בפלטפורמה בשם **vercel** המאפשרת ליצור אתרי אינטרנט (לא רק – היא עושה גם דברים אחרים). בחרתי בה כי היא תומכת בצורה טובה מאוד ב- Node.js והיא פשוטה יחסית לשימוש.

דבר ראשון, נרשמתי לאתר **vercel.com** ויצרתי משתמש במערכת (הפלטפורמה מקשרת בינה לבין GitHub). לאחר מכן הייתי צריך לעלות את הקוד שיצרתי, וכך עשיתי.

נכנסתי ל- cmd במיקום קבצי האתר שיצרתי והורדתי חבילה בשם **vercel** בעזרת הפקודה הבאה:

npm install -g vercel

לאחר הורדת החבילה, התחברתי אל המשתמש vercel שיצרתי, בעזרת הפקודה הבאה:

vercel login

```
C:\Users\MSI\Desktop\NETWORK\Final Exercise\step D - the website>vercel login
Vercel CLI 37.1.1
? Log in to Vercel Continue with GitHub
> Success! GitHub authentication complete for elielmonf@gmail.com
Congratulations! You are now logged in. In order to deploy something, run `vercel`.
💡 Connect your Git Repositories to deploy every branch push automatically (https://vercel.link/git).
```

לאחר ההתחברות, צריך ליצור פרויקט חדש. אם הייתי יוצר פרויקט חדש באותו רגע, האתר לא היה עובד כמו שצריך. דבר זה קרה (כן – יצרתי פרויקט לפני. דבר שתקע אותי במשך שעות) מפני שאנחנו משתמשים ב- Node.js ליצירת השרת שלנו, ולכן הוספתי קובץ חדש בשם **vercel.json** שתפקידו להפעיל את קובץ השרת שיצרנו (server.js).

קוד ה- **vercel.json**

```
{
  "version": 2,
  "builds": [
    { "src": "server.js", "use": "@vercel/node" }
  ],
  "routes": [
    { "src": "/(.*)", "dest": "server.js" }
  ]
}
```

כעת אפשר ליצור את הפרויקט על המשתמש ולהעלות את הקבצים. עשיתי זאת בעזרת הפקודה **vercel**, ואחר כך עברתי על מספר שלבים ליצירת הפרויקט.

```
C:\Users\MSI\Desktop\NETWORK\Final Exercise\step D - the website>vercel
Vercel CLI 37.1.1
? Set up and deploy “~\Desktop\NETWORK\Final Exercise\step D - the website”? yes
? Which scope do you want to deploy to? eliel's projects
? Link to existing project? no
? What's your project's name? bobbase
? In which directory is your code located? ./
Local settings detected in vercel.json:
No framework detected. Default Project Settings:
- Build Command: 'npm run vercel-build' or 'npm run build'
- Development Command: None
- Install Command: 'yarn install', 'pnpm install', 'npm install', or 'bun install'
- Output Directory: 'public' if it exists, or '.'
? Want to modify these settings? no
🔗 Linked to eliels-projects-a6d31f90/bobbase (created .vercel and added it to .gitignore)
🔗 Inspect: https://vercel.com/eliels-projects-a6d31f90/bobbase/5notmwUtJmVnBRjcRyRUDzhAcyxJ [2s]
✅ Production: https://bobbase-omsh8rpd-eliels-projects-a6d31f90.vercel.app [2s]
📦 Deployed to production. Run 'vercel --prod' to overwrite later (https://vercel.link/2F).
💡 To change the domain or build command, go to https://vercel.com/eliels-projects-a6d31f90/bobbase/settings
```

האתר כעת נימצא ב- Preview. כלומר שהוא עדיין לא נמצא באפשרות גישה של כולם. כדי להעביר אותו לאינטרנט ולקבל קישור ציבורי הרצתי את הפקודה הבאה:


vercel --prod

```
C:\Users\MSI\Desktop\NETWORK\Final Exercise\step D - the website>vercel --prod
Vercel CLI 37.1.1
🔗 Inspect: https://vercel.com/eliels-projects-a6d31f90/bobbase/C4xQqWdQp7xKNfRTLgpitsN9tUkx [1s]
✅ Production: https://bobbase-9w37he7fs-eliels-projects-a6d31f90.vercel.app [1s]
! Due to 'builds' existing in your configuration file, the Build and Development Settings defined in your Project Settings will not apply. Learn More: https://vercel.link/unused-build-settings
```

עכשיו האתר באינטרנט, ואם נתחבר לאתר דרך המשתמש שיצרתי נוכל לקבל את ה- URL הסופי של האתר שסוף סוף יצרתי ([לביקור קצר](#)).

Production Deployment

The deployment that is available to your visitors.



Deployment

bobbase-9w37he7fs-eliels-projects-a6d31f90.vercel.app

Domains

[bobbase.vercel.app](#) [bobbase-eliels-projects-a6d31f90.vercel.app](#) +1

Status

Created

● Ready 2h ago by eliel-monfort

Source

<> View code

>_ vercel deploy

כעת נעבור לעיקר השלב – שימוש ב- Gobaster tool למציאת הנתבים המוסתרים המכילים את ההסנפה ומפתח לפיענוח ההסנפה.

כדי להשתמש בפקודה עלינו להוריד למחשב שלושה דברים:

(1) **Go** – שפת תכנות.

(2) **Gobuster** – הפקודה עצמה.

השתמשתי בפקודה **go install github.com/OJ/gobuster/v3@latest** להורדת הפקודה.

(3) **Wordlist** – רשימת מילים שהפקודה תעבור עליה ותבצע brute force.

את רשימת המילים הורדתי מהקישור הבאה,

<https://github.com/digination/dirbuster-ng/blob/master/wordlists/common.txt>

את רשימת המילים שמרתי בקובץ **common.txt**.

לאחר כל ההורדות, והוספת נתיבי הקבצים של Go והסקריפט של Gobuster ל-PATH כדי שנוכל להריץ אותם בכל מקום, הרצתי את הפקודה על האתר שלי בצורה הבאה:

gobuster dir -u https://bobbbase.vercel.app -w C:\Users\MSI\go\bin\common.txt

לאחר הרצת הפקודה יכולתי לראות את הנתיבים המוסתרים שיצרתי מוחזרים (איזו התרגשות!).

```
C:\Users\MSI>gobuster dir -u https://bobbbase.vercel.app -w C:\Users\MSI\go\bin\ common.txt
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:             https://bobbbase.vercel.app
[+] Method:          GET
[+] Threads:         10
[+] Wordlist:         C:\Users\MSI\go\bin\
[+] Negative Status codes: 404
[+] User Agent:      gobuster/3.6
[+] Timeout:         10s
=====
Starting gobuster in directory enumeration mode
=====

Error: error on running gobuster: failed to get number of lines: read C:\Users\MSI\go\bin\: Incorrect function.

C:\Users\MSI>gobuster dir -u https://bobbbase.vercel.app -w C:\Users\MSI\go\bin\common.txt
=====
Gobuster v3.6
by OJ Reeves (@TheColonial) & Christian Mehlmauer (@firefart)
=====
[+] Url:             https://bobbbase.vercel.app
[+] Method:          GET
[+] Threads:         10
[+] Wordlist:         C:\Users\MSI\go\bin\common.txt
[+] Negative Status codes: 404
[+] User Agent:      gobuster/3.6
[+] Timeout:         10s
=====
Starting gobuster in directory enumeration mode
=====
/Downloads      (Status: 301) [Size: 185] [--> /Downloads/]
/backup          (Status: 301) [Size: 179] [--> /backup/]
/security        (Status: 301) [Size: 183] [--> /security/]
/uploads         (Status: 301) [Size: 181] [--> /uploads/]
Progress: 1942 / 1942 (100.00%)
=====
Finished
=====
```

כמובן שאם אמשיך לחקור אוכל למצוא את המשכי הנתיבים, וכאשר אכניס את התוצאות שיצאו לי אוכל למצוא את ההסנפה והמפתח (שאצור בשלב הבאה). כעת סיימנו שלב נוסף.

שלב 5:

המטרה:

יצירת הסנפת TLS עם התחברות לאתר שיצרתי על ידי שם המשתמש והסיסמא. הוספת קובץ ההסנפה והמפתח ההצפנה הסימטרי לתוך האתר.

הביצוע:

פתחתי Wireshark והתחלתי הסנפה. בזמן ההסנפה התחברתי לאתר בעזרת שם המשתמש והסיסמא. לאחר סיום ההתחברות, עצרתי את ההסנפה ושמרתי אותה ואת המפתחות הסימטריים שנוצרו לי בקובץ **sslkeylog.txt**. כמובן שזיהיתי את המפתחות המתאימים לפי ה- Client Random ושמרתי אותם בקובץ נפרד.

את קובץ ההסנפה שמרתי ב- Google Drive ואת הקישור שלו הכנסתי לאתר בנתיב **Downloads/network/** ואת המפתחות גם הכנסתי לאתר בנתיב **security/key/**. וכך סיימנו את השלב האחרון. היידיה.

רפלקציה

במהלך פיתוח ה- CTF חזרתי לא מעט פעמים על נושאים שלמדנו במהלך הקורס, אך הדבר הנפלא ביותר הוא שלמדתי הרבה מאוד דברים חדשים, נושאים שרק שמעתי עליהם אך בפרויקט יישמתי אותם. אני חושב שהתרגיל פתח לי את העיניים והרחיב את הבנתי על כל מיני נושאים שלא הבנתי עד הסוף. נהניתי מאוד!