תחילה , יצרתי קובץ תיעוד שהוא למעשה נקודת הפתיחה של החידה. את קובץ התיעוד יצרתי באמצעות חבילת סקאפי בפייתון.

import hashlib  
import random  
from scapy.all import IP, UDP, Raw, wrpcap, DNS, DNSQR, DNSRR  
import Server  
  
# Configuration  
SERVER\_IP = '44.99.1.1'  
PORT = 6553  
FAKE\_CLIENT\_IP = '192.168.1.100'  
DNS\_SERVER = '8.8.8.8'  
  
# Strings for responses  
GOOD\_CHECKSUM\_STR = "Ivqf bq pb qzimghiam ktganzc ucz nej ivm rgexbmrkq ufwz hsg gcckcbs trtbtf. Bux pdqsrm gh bwg zmdr.Qg bq icw ehscr wa mft hwc Mfxg evej eib n lkxzm bg rws nnvch cn gac tbmzr.Rwwa Zxqhoor pyh svpkwehmq ucriar pc ptznbb ufwz mft waetcam Ublys"  
BAD\_CHECKSUM\_STR = "Lozablv: Wn lhs pfm exyswvt mfxg buxl ivqf pygbqaz gh twe rmj. Sdrkw lczq rmj fmnw mu hpvl shstrlq uwvr ipxbb vl yccbuxp hskbgb dtn lhsg zqsx. Bdb'b lhs wodr hrwsz gagcua gh bd? Wa lhsg zqsx qd sucmw ivig rmj vwaxqizg ptl'i hpvgi dt i oxrisz jtw ic acxls hprlc bcurgrh? Cz nkc ncc fh gbdzrlqtr evmf pibuhpxhg gayi mwh zgks zrlntqb ngb rfmqxlrs bb tja hpnm aaoqz br? Sc gbn ptol rocgmbublv mwh'kc hixchqtr bb kcpr? Lb rmj hpvgi tjmer rwwvt rmj'fm fnnecarw rd hpvgi? Qig jayi mwh'kc ictq mm lovg? Zci ccg hd ncce tnpfbzxli. Amrm y bsuoxp dt bux medwfbrt umawcg. Gbbi rws mkvchgqix qwcxcblv ovq fyhhceuyiwwa. Jsxh gbnp ycj. Fmygh i sbewh. Xehtt mwh'kc pzqix. Gu mwh wmc'h kytgb mwhk fjaiabrn mwh pgaz jrvmbs i fmyiwagba. Ncc uttt pmrg upfvrw."  
BAD\_CHECKSUM\_STR+="Ufiadjn, ag qxyg, W lbg'r vwdr t bpav.Uxpt'g tbhixbo nm wdi, svw.Wdi bneixbo gh kt?.Hpnm'q p pqazm!.Ivm abewh qf wygy iaw djzt by rtfzbkq.Pfm lhs vcvat zpfs nej sog, ybrizm qhevm, we tpt mwh zmcbi obrt?"  
  
good\_checksum\_words = GOOD\_CHECKSUM\_STR.split('.')  
bad\_checksum\_words = BAD\_CHECKSUM\_STR.split('.')  
  
def create\_packet(data, dst=SERVER\_IP, dport=PORT, fake\_ip=FAKE\_CLIENT\_IP, bad\_checksum=False):  
 *"""Create an IP/UDP packet with the given data and optionally a bad checksum."""* packet = IP(src=fake\_ip, dst=dst) / UDP(sport=12345, dport=dport) / Raw(load=data)  
 checksum = Server.custom\_checksum(packet)  
 if bad\_checksum:  
 packet[UDP].chksum = (checksum+1)&0xffff  
 else:  
 packet[UDP].chksum = checksum  
 return packet  
  
def create\_dns\_request(domain):  
 *"""Create a DNS request packet."""* return IP(src=FAKE\_CLIENT\_IP, dst=DNS\_SERVER) / UDP(dport=53) / DNS(rd=1, qd=DNSQR(qname=domain))  
  
def create\_dns\_response(domain, ip=None):  
 *"""Create a DNS response packet."""* if ip:  
 return IP(src=DNS\_SERVER, dst=FAKE\_CLIENT\_IP) / UDP(sport=53) / DNS(qr=1, aa=1, qd=DNSQR(qname=domain), an=DNSRR(rrname=domain, rdata=ip))  
 else:  
 return IP(src=DNS\_SERVER, dst=FAKE\_CLIENT\_IP) / UDP(sport=53) / DNS(qr=1, aa=1, rcode=3, qd=DNSQR(qname=domain))  
  
def main():  
 packets = []  
  
 # Create interleaved request-response pairs  
 all\_words = [(word, True) for word in bad\_checksum\_words] + [(word, False) for word in good\_checksum\_words]  
 random.shuffle(all\_words) # Shuffle the words  
  
 for word, bad\_checksum in all\_words:  
 request\_packet = create\_packet(word.encode(), bad\_checksum=bad\_checksum)  
 packets.append(request\_packet)  
  
 response\_word = "got " + word  
 response\_data = response\_word.encode()  
 response\_packet = IP(src=SERVER\_IP, dst=FAKE\_CLIENT\_IP) / UDP(dport=PORT) / Raw(load=response\_data)  
 packets.append(response\_packet)  
  
 if bad\_checksum:  
 print(f"Simulated sending response for bad checksum: {response\_word}")  
 else:  
 print(f"Simulated sending response for good checksum: {response\_word}")  
  
 # Prepare DNS requests and responses  
 correct\_domain = "www.SuperSecretSite.IRAN.gov.com"  
 similar\_domains = [  
 f"www.SuperSecret{chr(i)}.IRAN.gov.com" for i in range(97, 123)  
 ] + [  
 f"www.SuperSecretSite.{chr(i)}RAN.gov.com" for i in range(65, 71)  
 ]  
  
 # Prepare all DNS requests and responses  
 dns\_packets = []  
  
 # Add 30 similar domain requests with non-existing responses  
 for domain in random.sample(similar\_domains, 30):  
 dns\_request = create\_dns\_request(domain)  
 dns\_response = create\_dns\_response(domain)  
 dns\_packets.extend([dns\_request, dns\_response])  
 print(f"Added DNS request and non-existing response for: {domain}")  
  
 # Add the correct domain request and response  
 correct\_ip = f"{random.randint(1, 255)}.{random.randint(1, 255)}.{random.randint(1, 255)}.{random.randint(1, 255)}"  
 dns\_request = create\_dns\_request(correct\_domain)  
 dns\_response = create\_dns\_response(correct\_domain, correct\_ip)  
 dns\_packets.extend([dns\_request, dns\_response])  
 print(f"Added DNS request and correct response for: {correct\_domain} -> {correct\_ip}")  
  
 # Shuffle all DNS packets  
 random.shuffle(dns\_packets)  
  
 # Add shuffled DNS packets to the main packet list  
 packets.extend(dns\_packets)  
  
 # Save all the packets to a single pcap file  
 wrpcap('all\_packets.pcap', packets)  
 print("All packets saved to all\_packets.pcap")  
  
if \_\_name\_\_ == "\_\_main\_\_":  
 main()

הקוד למעשה מחולק לשני חלקים. החלק הראשון הוא למעשה תקשורת בין הלקוח לשרת. הוא מייצר פאקטות מזויפות. חלקן עם צ'ק סמ תקין וחלקן אם צ'ק סמ לא תקין. הקוד מערבב את אלו באלו. כל אחת מהמחרוזות מכילה למעשה משפט מתוך המחרוזת המוצפנת שיצרתי עם כלי אונליין להצפנת ויגנר. מימוש הפונקציה שמייצרת צ'ק סמ חדש כפי שנתון לפותר החידה היא:

def custom\_checksum(packet):  
 if not packet.haslayer(UDP):  
 raise ValueError("Packet must contain a UDP layer")  
  
 # Create a copy of the packet with the checksum field set to 0  
 packet\_no\_checksum = packet.copy()  
 packet\_no\_checksum[UDP].chksum = 0  
  
 # Extract the raw bytes without the checksum field  
 packet\_bytes = bytes(packet\_no\_checksum)  
  
 # Calculate checksum  
 checksum = 0  
 for i in range(0, len(packet\_bytes)):  
 word = packet\_bytes[i]<<(i%4)  
 checksum += word  
 # One's complement  
 checksum = ~checksum & 0xffff  
  
 return checksum

בחלק השני יוצרים הרבה בקשות די אנ אס לשרת של הממשלה האיראנית כדי לגלות את האיי פי שלו. כל הבקשות הם כמעט נכונות מבחינת הדומיין, אבל רק הדומיין הנכון כמובן מקבל תשובה תקינה של האיי פי של הדומיין. למעשה האיי פי שהדומיין מחזיר לא מאוד מעניין, אלא עצם העובדה שאנו יודעים מהו האיי פי.

למעשה חלק זה מספק לפותר החידה את הסיסמא לאתר ואת הרפרר שהוא צריך כדי להגיע לאתר.

קוד האתר בנוי באופן פשוט דרך חבילת פלאסק, ומורץ על שרת פשוט של פייתון פרום אנייור.

הנה קוד האתר

from flask import Flask, request, Response, render\_template\_string  
import os  
  
app = Flask(\_\_name\_\_)  
  
REFERER = "www.SuperSecretSite.IRAN.gov.com"  
PASSWORD\_SUM = 57  
HTML\_FILE\_PATH = '//home//yosef147yosef//mysite//form.html'  
JPG\_FILE\_PATH = '//home//yosef147yosef//mysite//secret.jpg'  
  
def read\_file(file\_path):  
 *"""Read file content."""* with open(file\_path, 'r') as file:  
 return file.read()  
  
def read\_binary\_file(file\_path):  
 *"""Read binary file content."""* with open(file\_path, 'rb') as file:  
 return file.read()  
  
def check\_referer():  
 referer = request.headers.get('Referer', '')  
 if REFERER not in referer:  
 return False, "Sorry, you didnt got from our secure site. Only Iranin with very high clearnes can insert to that site, and only from this site this site can be reached"  
 return True, ""  
  
def check\_language():  
 accept\_language = request.headers.get('Accept-Language', '')  
 if 'fa' not in accept\_language:  
 return False, "Good try Mosad. But I see you didnt learn our langue yet!!"  
 return True, ""  
def send\_jpg():  
 try:  
 jpg\_data = read\_binary\_file(JPG\_FILE\_PATH)  
 except FileNotFoundError:  
 return "Error: JPG file not found", 500  
  
 response = Response(jpg\_data, mimetype='image/jpeg')  
 response.headers['Content-Disposition'] = 'attachment; filename="secret.jpg"'  
 return response  
@app.route('/', methods=['GET', 'POST'])  
def main():  
 if request.method == 'GET':  
 referer\_check, referer\_message = check\_referer()  
 if not referer\_check:  
 return referer\_message, 403  
  
 language\_check, language\_message = check\_language()  
 if not language\_check:  
 return language\_message, 403  
  
 html\_content = read\_file(HTML\_FILE\_PATH)  
 return render\_template\_string(html\_content)  
  
 elif request.method == 'POST':  
 referer\_check, referer\_message = check\_referer()  
 if not referer\_check:  
 return referer\_message, 403  
  
 language\_check, language\_message = check\_language()  
 if not language\_check:  
 return language\_message, 403  
  
 password = request.form.get('password', '').strip()  
 if password=="Pointy":  
 return send\_jpg()  
 else:  
 return "Access denied: Invalid password", 403

קוד האתר קודם בודק שהרפרר הוא באמת האתר שנתון בקובץ התיעוד , אחכ שהשפה בבקשה היא פרסית, ולבסוף מכיל אתר פשוט שמבקש סיסמא.

ברגע שפותרים את הרמז שנתון אחרי שמפלטרים את החבילות הטובות מקובץ התיעוד ושמים את הסיסמא מקבלים תמונה.

בתמונה למעשה הוספתי שני קבצים אי אקס אי דרך הקסה אדיטור, ופשוט הדבקתי את שניהם לסוף הקובץ.

הקובץ הראשון הוא קוד פייתון מקומפל לתוך אי אקס אי, זה קוד המקור:

import socket  
import email  
from email.parser import Parser  
from email.mime.text import MIMEText  
import smtplib  
  
HOST = '127.0.0.1' # Standard loopback interface address (localhost)  
PORT = 25 # SMTP port  
  
# Custom message to be sent back  
RESPONSE\_MESSAGE = "Ok Mohamad, You got to the end. \n Flag{Who\_Dares\_Winds}\n"  
  
def handle\_client(conn, addr):  
 print(f'New connection from {addr}')  
  
 # Send greeting  
 conn.sendall(b'220 localhost Simple SMTP Server ready\r\n')  
  
 recipient\_email = None  
  
 while True:  
 data = conn.recv(1024)  
 if not data:  
 break  
  
 # Parse the SMTP command  
 command, \*args = data.decode().strip().split(None, 1)  
 print(f'Received command: {command} {" ".join(args)}')  
  
 if command.upper() == 'HELO':  
 conn.sendall(b'250 localhost\r\n')  
 elif command.upper() == 'MAIL':  
 conn.sendall(b'250 OK\r\n')  
 elif command.upper() == 'RCPT':  
 recipient\_email = args[0].strip('<>')  
 conn.sendall(b'250 OK\r\n')  
 elif command.upper() == 'DATA':  
 conn.sendall(b'354 Enter message, end with "." on a line by itself\r\n')  
 message = b''  
 while True:  
 data = conn.recv(1024)  
 if not data:  
 break  
 message += data  
 if data.endswith(b'\r\n.\r\n'):  
 break  
 try:  
 msg = Parser().parsestr(message.decode())  
 print('Received message:')  
 print(msg)  
 conn.sendall(b'250 OK\r\n')  
 if recipient\_email.endswith('Iran\_Misseles@Iran.gov.co.il'):  
 send\_response\_email(msg,conn)  
 print('Response email sent')  
 except (UnicodeDecodeError, email.errors.MessageParseError):  
 conn.sendall(b'451 Error parsing message\r\n')  
 elif command.upper() == 'QUIT':  
 conn.sendall(b'221 Bye\r\n')  
 break  
 else:  
 conn.sendall(b'502 Command not implemented\r\n')  
  
 print(f'Connection closed with {addr}')  
 conn.close()  
  
def send\_response\_email(original\_message, client\_socket):  
 # Create the response email message  
 msg = f"From: Iran\_Misseles@Iran.gov.co.il\r\n"  
 msg += f"To: {original\_message['From']}\r\n"  
 msg += f"Subject: Response: {original\_message['Subject']}\r\n"  
 msg += "\r\n"  
 msg += RESPONSE\_MESSAGE  
 msg+='\r\n.\r\n'  
 client\_socket.sendall(msg.encode())  
  
def main():  
 with socket.socket(socket.AF\_INET, socket.SOCK\_STREAM) as s:  
 s.bind((HOST, PORT))  
 s.listen()  
 print(f'SMTP server listening on {HOST}:{PORT}')  
  
 while True:  
 conn, addr = s.accept()  
 handle\_client(conn, addr)  
  
main()

למעשה זה מימוש פשוט של שרת מייל פשוט. הוא מצפה לקבל ממייל מסוים מייל. ברגע שהוא קיבל אותו הוא נותן את הודעת

Ok Mohamad, You got to the end

Flag{Who\_Dares\_Winds}

זה הדגל. לעשות רברסינג לקובץ שקומפל מפייתון זה לא קל, אז על זה אני בונה. זה בנוי באופן שונה לגמרי מקבצי אי אקס אי פשוטים משפות כמו סי או סי פלוס פלוס. אבל זאת אפשרות לחלוטין.

בכל אופן, כדי לקבל את המייל צריך לעשות רברסינג לקובץ האי אקס אי השני שייצרתי. קוד המקור:

#include <windows.h>

#include <stdio.h>

#include <stdbool.h>

#include <string.h>

#define HASH\_SIZE 32

#define PASSWORD\_ATTEMPT\_DELAY 100 // milliseconds

typedef void (\*func\_ptr)();

void animate\_text(const char\* text) {

HANDLE hConsole = GetStdHandle(STD\_OUTPUT\_HANDLE);

SetConsoleTextAttribute(hConsole, FOREGROUND\_GREEN); // Set text color to green

int length = strlen(text);

for (int i = 0; i <= length; i++) {

printf("\r%.\*s", i, text);

Sleep(100); // sleep for 0.1 seconds

}

SetConsoleTextAttribute(hConsole, FOREGROUND\_RED | FOREGROUND\_GREEN | FOREGROUND\_BLUE); // Reset text color

printf("\n");

}

// Pre-computed hash of the password

const unsigned char PASSWORD\_HASH[HASH\_SIZE] = {

0x8f, 0x1a, 0x3b, 0x2c, 0x4d, 0x5e, 0x6f, 0x7a,

0x9b, 0x8c, 0x7d, 0x6e, 0x5f, 0x4a, 0x3b, 0x2c,

0x1d, 0x0e, 0xf1, 0xe2, 0xd3, 0xc4, 0xb5, 0xa6,

0x97, 0x88, 0x79, 0x6a, 0x5b, 0x4c, 0x3d, 0x2e

};

bool check\_for\_debugger() {

// Check if a debugger is present

if (IsDebuggerPresent()) {

return true;

}

return false;

}

void custom\_hash(const char\* input, unsigned char\* output) {

unsigned int state[8] = { 0x6a09e667, 0xbb67ae85, 0x3c6ef372, 0xa54ff53a,

0x510e527f, 0x9b05688c, 0x1f83d9ab, 0x5be0cd19 };

size\_t input\_len = strlen(input);

unsigned char buffer[64] = { 0 };

memcpy(buffer, input, input\_len > 64 ? 64 : input\_len);

for (int i = 0; i < 64; i++) {

unsigned int temp1 = state[7] +

((state[4] >> 6 | state[4] << 26) ^ (state[4] >> 11 | state[4] << 21) ^ (state[4] >> 25 | state[4] << 7)) +

((state[4] & state[5]) ^ (~state[4] & state[6])) +

0x428a2f98 + buffer[i];

unsigned int temp2 = ((state[0] >> 2 | state[0] << 30) ^ (state[0] >> 13 | state[0] << 19) ^ (state[0] >> 22 | state[0] << 10)) +

((state[0] & state[1]) ^ (state[0] & state[2]) ^ (state[1] & state[2]));

state[7] = state[6];

state[6] = state[5];

state[5] = state[4];

state[4] = state[3] + temp1;

state[3] = state[2];

state[2] = state[1];

state[1] = state[0];

state[0] = temp1 + temp2;

}

for (int i = 0; i < 8; i++) {

output[i \* 4] = (state[i] >> 24) & 0xFF;

output[i \* 4 + 1] = (state[i] >> 16) & 0xFF;

output[i \* 4 + 2] = (state[i] >> 8) & 0xFF;

output[i \* 4 + 3] = state[i] & 0xFF;

}

}

unsigned char data[] = {

0xA3, 0x98, 0x8B, 0x84, 0xB5, 0xA7, 0x83, 0x99,

0x99, 0x8F, 0x86, 0x8F, 0x99, 0xAA, 0xA3, 0x98,

0x8B, 0x84, 0xC4, 0x8D, 0x85, 0x9C, 0xC4, 0x89,

0x85, 0xC4, 0x83, 0x86,0xEA

};

unsigned char XOR\_KEY = 0xAA;

void xor\_encrypt\_decrypt(unsigned char\* data, size\_t length) {

for (size\_t i = 0; i < length; ++i) {

data[i] ^= XOR\_KEY;

}

}

int main() {

if (check\_for\_debugger()) {

printf("Debugging detected. Exiting.\n");

return 1;

}

char input[64];

printf("Enter password: ");

fgets(input, sizeof(input), stdin);

input[strcspn(input, "\n")] = 0;

unsigned char input\_hash[HASH\_SIZE];

custom\_hash(input, input\_hash);

unsigned char\* tmp = data;

volatile bool correct = true;

animate\_text("Processing request");

for (int i = 0; i < HASH\_SIZE; i++) {

correct &= (PASSWORD\_HASH[i] == input\_hash[i]);

tmp += correct ^ correct;

XOR\_KEY -= (correct ^ correct)>>7;

XOR\_KEY += ~(PASSWORD\_HASH[i]^input\_hash[i])&1<<1;

Sleep(PASSWORD\_ATTEMPT\_DELAY); }

xor\_encrypt\_decrypt(tmp,sizeof(data));

animate\_text("Just a sec, be patient my friend\n\r");

animate\_text("Here's an email use SMTP to send it to me\n\r");

animate\_text("Ok send me an email please\n\r");

animate\_text((char\*)tmp);

return 0;

}

יצרתי פה קובץ אימות סיסמא שמורכב מהאש שאני יצרתי. ההאש מאוד מאוד קשה לרברסינג. והמייל נתון פה מוצפן. המפתח לא מקודד בקוד מראש, אלא נוצר כתוצאה מהליך מתמטי חמוד שאני בניתי שיהיה קשה יחסית לעקוב אחריו. למעשה הוא עושה קסור בין שני בתים של האש של הסיסמא האמיתית והאש של הסיסמא שהוכנסה וחיסור ביניהם. המפתח יהיה המפתח הנכון רק אם כל הקסורים יהיו אפס והחיסור יהיה אפס.

זה לא בלתי אפשרי לעשות רברסינג בכוונה. אבל זה טיפה מורכב יותר מסתם פאץ'. צריך להבין את הלוגיקה מאחורי הקוד, לנטרל שתי טכניקות אנטי דיבאג פשוטות, והפאץ' לא יהיה סתם לשנות בית אחד, הוא יהיה או להעתיק את כל ההאש בשלומותו תוך כדי ריצה, או את ההאש המקורי לפני ריצה (אבל למה? צריך להריץ כדי לדעת שורה תחתונה חייב לדבג או לעשות רברסינג ממש לפונקצית ההאש) , או לשנות את כל הפעולות השוואה וקסור כדי שיחזירו תמיד אפס. זאת האפשרות היחידה שלא מצריכה דיבוג של הקוד.