



# Introducere în criptografie și algoritmi de criptare, encodare sau hashing

Resurse utile pentru incepatori din UNbreakable România

[unbreakable.ro](https://unbreakable.ro)

|   |           |
|---|-----------|
| <b>Declinarea responsabilității</b>                                     | <b>4</b>  |
| <b>Introducere</b>  | <b>5</b>  |
| Ce este criptarea datelor?  | 5         |
| De ce este important sa invatam despre criptarea datelor?               | 5         |
| <b>Tipuri de tehnici și algoritmi criptografici</b>                     | <b>6</b>  |
| Criptografia clasica  | 6         |
| Exemplu - cifrul Caesar   | 6         |
| Exemplu practic   | 6         |
| Alti algoritmi criptografici clasici                                    | 7         |
| Criptografia moderna  | 7         |
| Exemplu - Criptare folosind cheie simetrică                             | 7         |
| <b>Tipuri de atacuri sau vulnerabilitati în sistemele criptografice</b> | <b>9</b>  |
| Atacurile pasive  | 9         |
| Atacurile active  | 9         |
| Exemple de atacuri asupra algoritmilor criptografici                    | 10        |
| <b>Resurse utile</b>  | <b>12</b> |
| <b>Librarii si unelte utile în rezolvarea exercițiilor</b>              | <b>13</b> |
| <b>Exerciții și rezolvări</b>   | <b>14</b> |
| Lost-message (ridicat)  | 14        |
| War-plan (usor)   | 25        |
| Secure-encryption (usor)  | 27        |
| Rsa-quiz (medium)   | 29        |
| Rgb (mediu)   | 33        |
| Crypto-twist (ușor)   | 34        |
| Corrupt-decryption (greu)   | 36        |
| Enigma (mediu)  | 36        |
| Prophet (hard)  | 38        |
| Communication (ușor)  | 41        |
| Elgamal-quiz (ușor)   | 43        |
| Entangle (greu)   | 45        |
| Surfing (mediu)   | 49        |



## Declinarea responsabilității

Aceste materiale și resurse sunt destinate exclusiv informării și discuțiilor, având ca obiectiv conștientizarea riscurilor și amenințarilor informatice dar și pregătirea unor noi generații de specialiști în securitate informatică.

Organizatorii și partenerii UNbreakable România nu oferă nicio garanție de niciun fel cu privire la aceste informații. În niciun caz, organizatorii și partenerii UNbreakable România, sau contractanții, sau subcontractanții săi nu vor fi răspunzători pentru niciun fel de daune, inclusiv, dar fără a se limita la, daune directe, indirecte, speciale sau ulterioare, care rezultă din orice mod ce are legătură cu aceste informații, indiferent dacă se bazează sau nu pe garanție, contract, delict sau altfel, indiferent dacă este sau nu din neglijență și dacă vătămarea a fost sau nu rezultată din rezultatele sau dependența de informații.

Organizatorii UNbreakable România nu aprobă niciun produs sau serviciu comercial, inclusiv subiectele analizei. Orice referire la produse comerciale, procese sau servicii specifice prin marca de servicii, marca comercială, producător sau altfel, nu constituie sau implică aprobarea, recomandarea sau favorizarea acestora de către UNbreakable România.

Organizatorii UNbreakable România recomandă folosirea cunoștințelor și tehnologiilor prezentate în aceste resurse doar în scop educațional sau profesional pe calculatoare, site-uri, servere, servicii sau alte sisteme informatice doar după obținerea acordului explicit în prealabil din partea proprietarilor.

Utilizarea unor tehnici sau unelte prezentate în aceste materiale împotriva unor sisteme informatice, fără acordul proprietarilor, poate fi considerată infracțiune în diverse țări.

În România, accesul ilegal la un sistem informatic este considerată infracțiune contra siguranței și integrității sistemelor și datelor informatice și poate fi pedepsită conform legii.

## Introducere

În fiecare secundă a zilei, informațiile sensibile, de la numerele cardurilor de credit până la dosare de sănătate, secrete militare sau, sunt transmise prin Internet. Datorită criptării, totul poate fi realizat în siguranță. Criptarea permite transferul datelor confidențiale dintr-o rețea în alta fără a fi compromise. **Când datele sunt criptate, acestea nu pot fi accesate și exploatate de către utilizatori neautorizați.**

## Ce este criptarea datelor?

Parolele, limitarea accesului la servere și servicii, firewall-urile și stocarea folosită dispozitive externe sunt toate mijloace adecvate de securizare a datelor, dar criptarea este metoda cea mai utilizată. Criptarea convertește mesajele text, e-mailurile și datele încărcările în text cifrat, ceea ce le face ilizibile de către oameni.

Procese de criptare utilizează algoritmi care convertesc datele în coduri atât de complexe încât cele mai puternice computere ar dura ani de zile pentru a le sparge. Numai o persoană sau un computer care are cheia corectă poate decripta rapid informațiile sau le poate readuce în forma originală. Cheia de decriptare este un alt algoritm care inversează procesul algoritmului de criptare.

## De ce este important să învățăm despre criptarea datelor?

Criptarea datelor ajută oamenii de sute de ani să transfere informații în mod sigur, fără ca un tert neautorizat să poată înțelege aceste informații.

De-a lungul timpului au apărut două provocări, ce continuă să stimuleze și astăzi:

- **Crearea unor algoritmi sau tehnici criptografice suficient de puternice** pentru ca datele originale să nu poată fi recuperate decât cei autorizați să le citească
- **Identificarea unor vulnerabilități în algoritmii criptografici** utilizați în comunicare dintre părți

## Tipuri de tehnici și algoritmi criptografici

Pe măsură ce metodele de criptografie au evoluat în timp, acestea pot fi separate în două categorii principale: criptografie clasică și modernă.

### Criptografia clasica

Spre deosebire de criptografia modernă care lucrează cu date binare, metodele clasice de criptografie au fost dezvoltate folosind doar cifre și litere.

Pe baza acestor informații avem două categorii principale:

- Cifrele monoalfabetice care utilizează o substituție fixă pe întregul mesaj
- Cifrele polialfabetice care folosesc o serie de substituții diferite pe întregul mesaj

### Exemplu - cifrul Caesar

Aceasta este una dintre cele mai simple metode de encodare, cunoscut și ca cifru prin substituție sau mutare/transpozitie (cunoscut în engleza ca shifted cipher), deoarece pentru a-l utiliza peste un mesaj, o literă este înlocuită / mutată cu alta bazată pe un număr fix de la 0 la 25 (numărul total de litere în limba engleză alfabet.)

#### Exemplu practic

Alex și Andreea sunt studenți. Pentru a-și proteja conținutul de e-mailuri de colegii curioși, au decis să adopte cifrul Caesar ca metodă de protecție, folosind „3” ca și cheie de schimbare. Folosind tabelul de mai jos putem vedea că:

HELLO = KHOOR

Avem această ieșire deoarece folosim 3 ca element de mutare. Asta înseamnă că litera „A”, mutată cu 3 poziții după aceasta, este litera „D”.

#### Alfabetul simplu

|   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |   |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| a | b | c | d | e | f | g | h | i | j | k | l | m | n | o | p | q | r | s | t | u | v | w | x | y | z |
| D | E | F | G | H | I | J | K | L | M | N | O | P | Q | R | S | T | U | V | W | X | Y | Z | A | B | C |

Alfabetul cifrului Caesar cu mutare 3

## Alti algoritmi criptografici clasici

În competitii de tip Capture the Flag, cum este și UNbreakable România, exista foarte mulți algoritmi criptografici clasici folosit, unii avand abordări netraditionale. Cu titlul, de exemplu lista de mai jos conține o serie de algoritmi criptografici clasici populari, după numele cunoscut în limba engleza:

- **Simple substitution Cipher** - Este un alt cifru monoalfabetic. Principala diferență între Caesar Cipher și acesta este că algoritmul "Simple substitution Cipher" acceptă mai mult decât o singură unitate de mutare. De aici puteti extrapola la: litere duble, triplete de litere și așa mai departe.
- **Playfair cipher** - un algoritm ce folosește o matrice de 5x5 pentru a înlocui fiecare litera din alfabet
- **Vigenere cipher** - o varianta imbunatatita a algoritmului Caesar, ce are câteva abordări printre care Vernam cipher sau One-time pad cipher
- **Algoritmi de transpunere**, algoritmi ce reordoneaza literele, fără a fi înlocuite:
  - Rail Fence cipher
  - Scytale cipher
  - Route cipher
  - Columnar Transposition
  - Double Transposition
  - Disrupted Transposition
- **Beaufort cipher** - tehnica ce are la baza Tabula recta, inventata in 1508 de Johannes Trithemius

## Criptografia moderna

Foarte diferită de criptografia clasică, criptografia modernă se bazează pe principii matematice, computer science, inginerie samd.

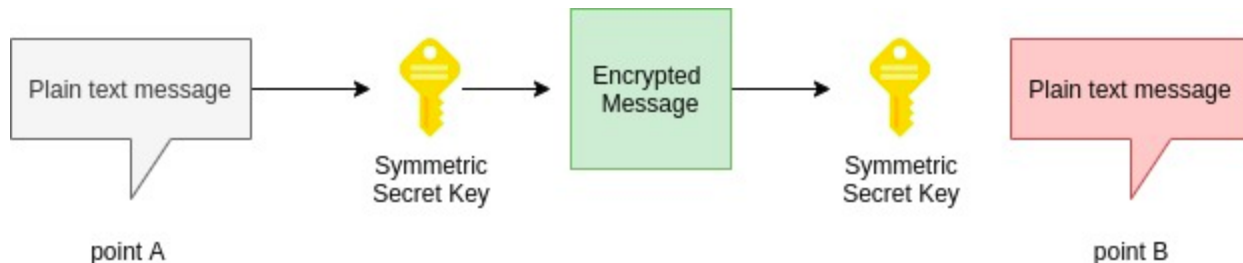
Principalele caracteristici ale criptografiei moderne sunt:

- Funcționează cu date binare
- Unele dintre metodele criptografiei moderne sunt imposibil de spart
- Toate procesele sunt automatizate, nu este necesară interacțiunea manuală, deoarece fiecare metodă de criptare este realizată de un computer

## Exemplu - Criptare folosind cheie simetrică

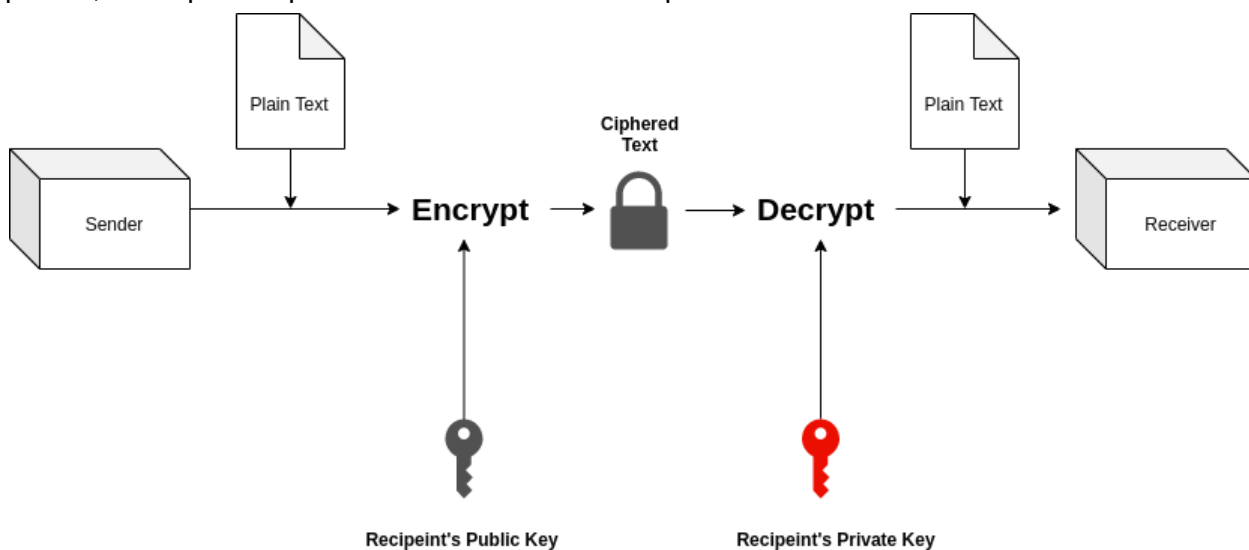
Principalul aspect al acestei metode este că folosește aceeași cheie pentru criptare și decriptare atunci când lucrează cu informații digitale.

Pentru ca două puncte (A & B) să comunice pe un canal criptat, trebuie să partajeze aceeași cheie secretă. Mai multe detalii despre modul de funcționare a criptării cheii simetrice pot fi găsite în imaginea de mai jos:



În competiții de tip Capture the Flag, cum este și UNbreakable România, sunt prezentate aplicații ce au algoritmi criptografici moderni ce sunt implementate greșit, au vulnerabilități cunoscute ce permit recuperarea mesajului original într-un termen rezonabil sau sunt reimplementări ale unor algoritmi consacrați, dar care au probleme de securitate. Printre cele mai populare algoritmi criptografici moderni, amintim de:

- **AES (Advanced Encryption Standard)**, cunoscut și sub numele de **Rijndael**, este un algoritm standardizat pentru criptarea simetrică, pe blocuri, folosit astăzi pe scară largă în aplicații și adoptat ca standard de organizația guvernamentală americană NIST.
- **PGP (Public Key Encryption)** - Asymmetric Key Encryption. Criptografia folosind cheie publică sau criptografie asimetrică este un sistem criptografic care folosește perechi de chei: chei publice (care pot fi cunoscute de alții) și chei private (care nu pot fi cunoscute niciodată de nimeni, cu excepția proprietarului). Generarea unor astfel de perechi de chei depinde de algoritmi criptografici care se bazează pe probleme matematice denumite funcții unidirecționale. Securitatea eficientă necesită păstrarea cheii private private; cheia publică poate fi distribuită fără a compromite securitatea datelor.





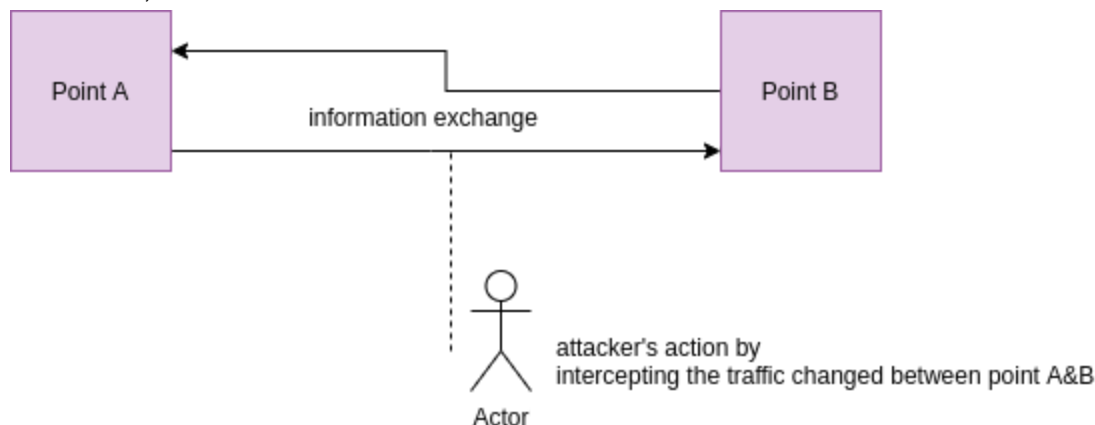
- **Funcțiile hash (clasă de funcții denumite în lucrări de specialitate și funcții de dispersie sau funcții de rezumat).** În informatică, procesul “hashing” este o practică obișnuită care implică conversia valorilor în alte date, conform algoritmilor de hash aleși. Funcțiile Hash sunt, de asemenea, numite funcții de compresie, deoarece procesul oferă o ieșire cu lungime fixă, chiar și în cazurile în care datele de intrare sunt foarte mari. Alții se referă la hash-uri ca funcții digest sau compresie deoarece, în general, reprezintă o valoare mai mică a unei date mai mari. Exemple: MD2, MD4, MD5, MD6, clasa SHA-0, clasa SHA-1, clasa SHA-2 (SHA224, SHA256, SHA384, SHA512) SHA-3, RIPEMD, WHIRLPOOL, CRC32

## Tipuri de atacuri sau vulnerabilitati în sistemele criptografice

Exista două categorii mari ce definesc tipurile de atacuri asupra sistemelor criptografice: atacuri pasive și atacuri active.

### Atacurile pasive

Scopul atacurilor pasive este de a obține acces neautorizat la date private / secrete fara implicarea activa in comunicatii. Putem numi acțiuni de atac pasiv, cum ar fi interceptarea traficului de rețea, ascultarea (ascultarea conversațiilor și canalelor de comunicare). Atacurile pasive funcționează astfel:



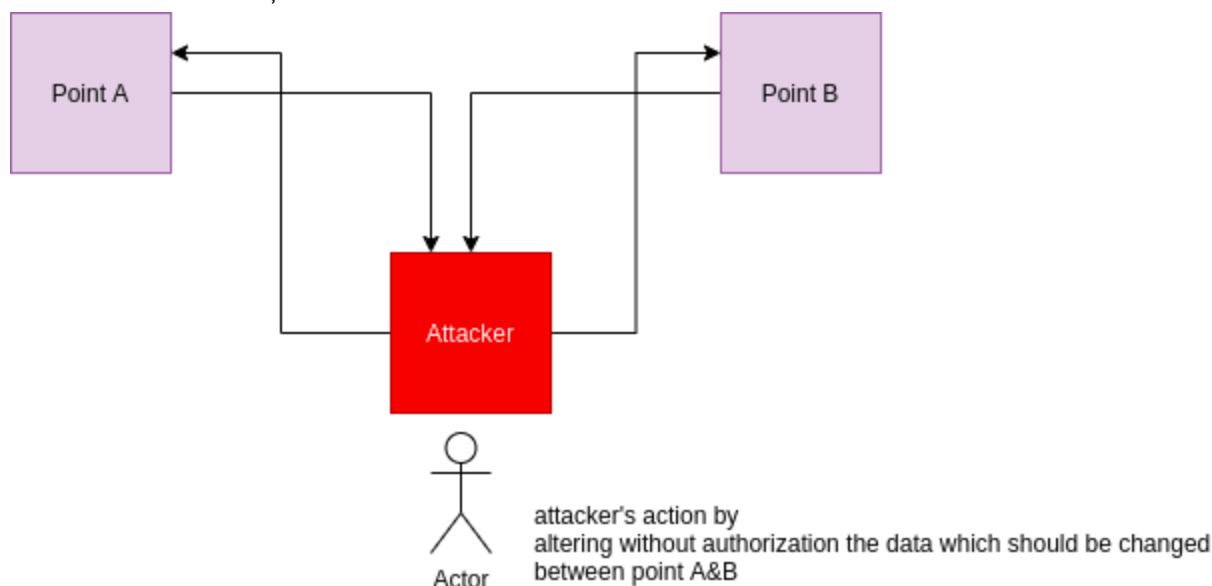
### Atacurile active

Spre deosebire de atacurile pasive în care actorii rău intenționați ascultă sau interceptează informațiile, atacurile active implică participarea directă a unui atacator.

În această categorie putem găsi atacuri precum:

- Modificarea datelor s-a schimbat între punctul A&B
- Modificări neautorizate, ștergere, creare de fișiere pe o anumită țintă
- Atacuri DoS & DDoS

Atacurile active funcționează astfel:



## Exemple de atacuri asupra algoritmilor criptografici

Fiecare algoritm criptografic poate avea vulnerabilitati din mai multe categorii, dar un bun punct de plecare pentru a va familiariza cu tehnicile folosite pentru a recupera sau intercepta datele criptate sunt acestea:

- Dictionary Attack
- Brute-force
- COA (Ciphred Only Attack)
- KPA (Known Plain Attack)
- CPA (Chosen Plaintext Attack)
- MIM (Man In Middle Attack)
- SCA (Side Channel Attack)
- Frequency Attacks
- Interpolation Attack
- Padding Oracle Attack
- Replay Attacks



## Resurse utile

- [https://www.tutorialspoint.com/cryptography\\_with\\_python/cryptography\\_with\\_python\\_quick\\_guide.htm](https://www.tutorialspoint.com/cryptography_with_python/cryptography_with_python_quick_guide.htm)
- <https://www.geeksforgeeks.org/playfair-cipher-with-examples/?ref=lbp>
- <https://secretpy.readthedocs.io/en/latest/>
- <https://pycryptodome.readthedocs.io/en/latest/src/features.html>

## Librarii si unelte utile în rezolvarea exercițiilor

- Python Crypto library
- <https://cryptii.com/>
- <https://www.quipqiup.com/>
- <https://www.alpertron.com.ar/ECM.HTM>
- <http://factordb.com/>

## Exerciții și rezolvări

### Lost-message (ridicat)

Concurs: UNbreakable #1 (2020)

Descriere:

My father was very paranoid with his communication and he encrypted every message he has ever sent.

This message was for me. Can you help me to decrypt the message ?

```
enc_message="FNFWCiZJGWWAWZTKYLLKDVNiWCVYViBYHxDiXFBEMiKYEZQMMPKN  
RiQXZVBQ"
```

Flag format = ctf{sha256(message)}

Goal: Perform a reverse engineering against the encryption algorithm and recover the original message.

The challenge was created by Bit Sentinel.

Rezolvare:

Pagina exercițiului conține și sursa programului folosit pentru a cripta mesajul:

```
#!/usr/bin/python3
import sys, random, binascii, hashlib, re, math, os
from string import ascii_uppercase as asc
from itertools import product as d

upper = {ascii:chr(ascii) for ascii in range(65,91)}
lower = {ascii:chr(ascii) for ascii in range(97,123)}
digit = {ascii:chr(ascii) for ascii in range(48,58)}
```

```
with open("text.txt", "r") as myfile:
    data = myfile.readline()

def premessage(text):

    text = text.replace("_", "Q")

    return text

def enc4(text, key, debug=False):

    r = [['\n' for i in range(len(text))
          for j in range(key)]

    dir_down = False
    row, col = 0, 0

    for i in range(len(text)):

        if (row == 0) or (row == key - 1):
            dir_down = not dir_down

        r[row][col] = text[i]
        col += 1

        if dir_down:
            row += 1
        else:
            row -= 1

    result = []
    for i in range(key):
        for j in range(len(text)):
            if r[i][j] != '\n':
                result.append(r[i][j])
```

```
return("".join(result))

def enc3(text, key):
    t=lambda x: x.upper().replace('J','I')
    s=[]
    for _ in t(key+asc):

        if _ not in s and _ in asc:

            s.append(_)

    m=[s[i:i+5] for i in range(0,len(s),5)]
    enc={row[i]+row[j]:row[(i+1)%5]+row[(j+1)%5] for row in m for i,j in d(range(5),repeat=2) if
    i!=j}
    enc.update({col[i]+col[j]:col[(i+1)%5]+col[(j+1)%5] for col in zip(*m) for i,j in
    d(range(5),repeat=2) if i!=j})
    enc.update({m[i1][j1]+m[i2][j2]:m[i1][j2]+m[i2][j1] for i1,j1,i2,j2 in d(range(5),repeat=4) if
    i1!=i2 and j1!=j2})
    l=re.findall(r'(.)(?!1)(.)?',".join(_ for _ in t(text) if _ in asc))

    return ".join(enc[a+(b if b else 'Z')] for a,b in l)

def enc2(string, key):
    for c in string:
        o = ord(c)
        if (o not in upper and o not in lower) or o in digit:
            yield o
        else:
            if o in upper and o + key % 26 in upper:
                yield o + key % 26
            elif o in lower and o + key % 26 in lower:
                yield o + key % 26.
            else:
                yield o + key % 26 -26
```



```
def enc1(msg, key):
    cipher = ""
    k_indx = 0
    msg_len = float(len(msg))
    msg_lst = list(msg)
    key_lst = sorted(list(key))
    col = len(key)
    row = int(math.ceil(msg_len / col))
    fill_null = int((row * col) - msg_len)
    msg_lst.extend('z' * fill_null)
    matrix = [msg_lst[i: i + col]
               for i in range(0, len(msg_lst), col)]

    for _ in range(col):
        curr_idx = key.index(key_lst[k_indx])
        cipher += ".join([row[curr_idx]
                          for row in matrix])
        k_indx += 1

    return cipher

cipher = enc1(enc3(enc4(premessage(data), 13), "recomanded"), "zxdfuypka")
cipher = "".join(map(chr, enc2(cipher, 35)))

print(cipher)
```

Textul inițial, a fost trecut prin patru funcții: enc1, enc2, enc3 și enc4. Pentru a putea recupera mesajul secret, va trebui să le inversăm pe toate.

Prima funcție, enc1, primește un text și o cheie de tip int. Va afișa apoi caracterele textului într-o matrice într-un mod determinabil din cheie și lungimea textului și le va citi pe coloane. Un mod interesant de a inversa această funcție este de a apela funcția enc1 cu un text de lungime egală care conține numere de la 1 la l și apoi de a reconstrui textul pe baza output-ului. În alte cuvinte, funcția enc1 creează o permutare a textului pe baza lungimii acestuia și a cheii. Folosind cheia de criptare și un text inițial cu caractere unice, putem determina permutarea care este identică pentru toate textele cu lungimi egale. Odată ce permutarea este determinată, aceasta poate fi inversată foarte ușor:

```
def dec1(msg, key):
    hack = [chr(_) for _ in range(len(msg))]
    hack = enc1(hack, key)
    dec = [" " for i in range(len(msg))]
    for i, v in enumerate(hack):
        pos = ord(v)
        dec[pos] = msg[i]
    return ".join(dec).rstrip('z')
```

Funcția enc2 este o implementare a cifrului Cezar și poate fi inversată foarte ușor folosind proprietățile acestui cifru:

```
def dec2(string, key):
    return enc2(string, 26 - (key % 26))
```

Funcția enc4 este asemănătoare din acest punct de vedere: reprezintă o implementare a cifrului Rail fence, ceea ce face posibilă aplicarea metodei folosite pentru inversarea funcției enc1:

```
def dec4(text, key, debug=False):
    text = list(text)
    dec = [0 for _ in range(len(text))]
    hack = ".join([chr(11 + i) for i in range(len(text))]) # \n is 10 and gets ignored
    hack = enc4(hack, key)
    for i, v in enumerate(hack):
        pos = ord(v) - 11
        dec[pos] = text[i]
    return ".join(dec)
```

Scriptul care decriptează mesajul secret poate fi găsit mai jos:

```
import sys, random, binascii, hashlib, re, math, os
from string import ascii_uppercase as asc
from itertools import product as d
import itertools, string

enc_message="FNFWCiZJGWWAWZTKYLLKDVNiWCvYViBYHXDiXFBEMiKYEZQMMPKN
```

```
RiQXZVBQ"
```

```
upper = {ascii:chr(ascii) for ascii in range(65,91)}  
lower = {ascii:chr(ascii) for ascii in range(97,123)}  
digit = {ascii:chr(ascii) for ascii in range(48,58)}
```

```
#with open("text.txt", "r") as myfile:  
#    data = myfile.readline()
```

```
def premessage(text):  
    text = text.replace("_", "Q")  
    return text
```

```
def postmessage(text):  
    text = text.replace("Q", "_")  
    return text
```

```
def dec4(text, key, debug=False):  
    text = list(text)  
    dec = [0 for _ in range(len(text))]  
    hack = ".join([chr(11 + i) for i in range(len(text))]) # \n is 10 and gets ignored  
    hack = enc4(hack, key)  
    for i, v in enumerate(hack):  
        pos = ord(v) - 11  
        dec[pos] = text[i]  
    return ".join(dec)
```

```
def enc4(text, key, debug=False):
```

```
    r = ["\n" for i in range(len(text))]  
        for j in range(key)]
```

```
    dir_down = False  
    row, col = 0, 0
```

```
    for i in range(len(text)):
```

```
if (row == 0) or (row == key - 1):
    dir_down = not dir_down

r[row][col] = text[i]
col += 1

if dir_down:
    row += 1
else:
    row -= 1

result = []
for i in range(key):
    for j in range(len(text)):
        if r[i][j] != '\n':
            result.append(r[i][j])
    return("".join(result))

double_counter = 0
def dec3(text, key):
    global double_counter
    t=lambda x: x.upper().replace('J','I')
    s=[]
    # asc = ascii_uppercase
    for _ in t(key+asc):
        if _ not in s and _ in asc:
            s.append(_)

    m=[s[i:i+5] for i in range(0,len(s),5)]
    enc={row[i] + row[j] : row[(i+1)%5] + row[(j+1)%5] for row in m for i,j in
itertools.product(range(5),repeat=2) if i!=j}
    enc.update({col[i]+col[j]:col[(i+1)%5]+col[(j+1)%5] for col in zip(*m) for i,j in
itertools.product(range(5),repeat=2) if i!=j})
    enc.update({m[i1][j1]+m[i2][j2]:m[i1][j2]+m[i2][j1] for i1,j1,i2,j2 in
itertools.product(range(5),repeat=4) if i1!=i2 and j1!=j2})

    rev_enc = {}
```

```
for k, v in enc.items():
    rev_enc[v] = k

l=re.findall(r'(.)(?:?!1)(.)?',".join([_ for _ in t(text) if _ in string.ascii_uppercase]))

dec = ""
for a, b in l:
    val = rev_enc[a + b]
    if val[1] == 'Z':
        double_counter += 1
        if double_counter == 1:
            val = val[0] + val[0]
        else:
            val = val[0]
    dec += val

return dec

def enc3(text, key):
    t=lambda x: x.upper().replace('J','I')
    s=[]
    for _ in t(key+asc):
        if _ not in s and _ in asc:
            s.append(_)

    m=[s[i:i+5] for i in range(0,len(s),5)]
    enc={row[i]+row[j]:row[(i+1)%5]+row[(j+1)%5] for row in m for i,j in d(range(5),repeat=2) if
    i!=j}
    enc.update({col[i]+col[j]:col[(i+1)%5]+col[(j+1)%5] for col in zip(*m) for i,j in
    d(range(5),repeat=2) if i!=j})
    enc.update({m[i1][j1]+m[i2][j2]:m[i1][j2]+m[i2][j1] for i1,j1,i2,j2 in d(range(5),repeat=4) if
    i1!=i2 and j1!=j2})

    l=re.findall(r'(.)(?:?!1)(.)?',".join([_ for _ in t(text) if _ in string.ascii_uppercase]))

    return ".join(enc[a+(b if b else 'Z')]) for a,b in l)

def dec2(string, key):
```

```
    return enc2(string, 26 - (key % 26))

def enc2(string, key):
    for c in string:
        o = ord(c)
        if (o not in upper and o not in lower) or o in digit:
            yield o
        else:
            if o in upper and o + key % 26 in upper:
                yield o + key % 26
            elif o in lower and o + key % 26 in lower:
                yield o + key % 26
            else:
                yield o + key % 26 - 26

def dec1(msg, key):
    hack = [chr(_) for _ in range(len(msg))]
    hack = enc1(hack, key)
    dec = [" " for i in range(len(msg))]
    for i, v in enumerate(hack):
        pos = ord(v)
        dec[pos] = msg[i]
    return ".join(dec).rstrip('z')

def enc1(msg, key):
    cipher = ""
    k_indx = 0
    msg_len = float(len(msg))
    msg_lst = list(msg)
    key_lst = sorted(list(key))
    col = len(key)
    row = int(math.ceil(msg_len / col))
    fill_null = int((row * col) - msg_len)
    msg_lst.extend('z' * fill_null)
    matrix = [msg_lst[i: i + col]
               for i in range(0, len(msg_lst), col)]
```

```
for _ in range(col):
    curr_idx = key.index(key_lst[k_idx])
    cipher += ".join([row[curr_idx]
                    for row in matrix])
    k_idx += 1

return cipher

# cipher = enc1(enc3(enc4(premessage(data), 13),"recomanded"), "zxdfiuyypka")
#cipher = "".join(map(chr, enc2(cipher, 35)))

#print(cipher)

print("Test time!")
test_string = "salut_pe_mine_nu_ma_cheama_george".replace("_", "q").upper()
print(dec4(enc4(test_string, 13), 13) == test_string)
print(dec3(enc3(test_string, "recomanded"), "recomanded")[:-1] == test_string)
test_string_enc2 = "".join(map(chr, enc2(test_string, 35)))
print("".join(map(chr, dec2(test_string_enc2, 35))) == test_string)
print(dec1(enc1(test_string, "zxdfiuyypka"), "zxdfiuyypka") == test_string)

flag = enc_message
flag = "".join(map(chr, dec2(flag, 35))) # ok
flag = dec1(flag, "zxdfiuyypka")
flag = dec3(flag, "recomanded")
flag = dec4(flag, 13) # ok
flag = postmessage(flag) # ok
print(flag)
```

Odată rulat, scriptul va face niște teste care ar trebui sa returneze valoarea **True** și apoi va decripta mesajul secret. Flag-ul este, de fapt, mesajul trecut prin funcția de hashing **md5**:

```
yakuhito@furry-catstation:~/ctf/unbr1/lost-message$ python solve.py
Test time!
True
True
```

```
True
True
KEEP_YOUR_COMUNICATION_ENCRYPTED_ALIENS_ARE_WATCHING
yakuhto@furry-catstation:~/ctf/unbr1/lost-message$ echo -n
KEEP_YOUR_COMUNICATION_ENCRYPTED_ALIENS_ARE_WATCHING | sha256sum
f5a2b03dedff103725131a2ce238bdc31b00accba79091237d566561cdfe6ec5 -
yakuhto@furry-catstation:~/ctf/unbr1/lost-message$
```

Rezolvare în engleză: <https://blog.kuhi.to/unbreakable-romania-1-writeup#lost-message>

## War-plan (usor)

Concurs: UNbreakable #2 (2020)

Descriere:

```
There is a hidden message in this file.

Find the message and win the war.

flag = ctf{sha256(message)}
```

Rezolvare:

Fisierul **wav** prezent e pagina exercițiului are 30 de minute si reprezinta un mesaj transmis prin codul morse. Putem folosi [acest site](#) pentru a recupera mesajul inițial:

```
THE BATTLE OF THE BULGE, ALSO KNOWN AS THE ARDENNES
COUNTEROFFENSIVE, WAS THE LAST MAJOR GERMAN OFFENSIVE CAMPAIGN ON
THE WESTERN FRONT DURING WORLD WAR II, AND TOOK PLACE FROM 16
DECEMBER 1944 TO 25 JANUARY 1945. IT WAS LAUNCHED THROUGH THE DENSELY
FORESTED ARDENNES REGION OF WALLONIA IN EASTERN BELGIUM, NORTHEAST
FRANCE, AND LUXEMBOURG, TOWARDS THE END OF THE WAR IN EUROPE. THE
OFFENSIVE WAS INTENDED TO STOP ALLIED USE OF THE BELGIAN PORT OF
ANTWERP AND TO SPLIT THE ALLIED LINES, ALLOWING THE GERMANS TO
ENCIRCLE AND DESTROY FOUR ALLIED ARMIES AND FORCE THE WESTERN ALLIES
TO NEGOTIATE A PEACE TREATY IN THE AXIS POWERS' FAVOR.
```



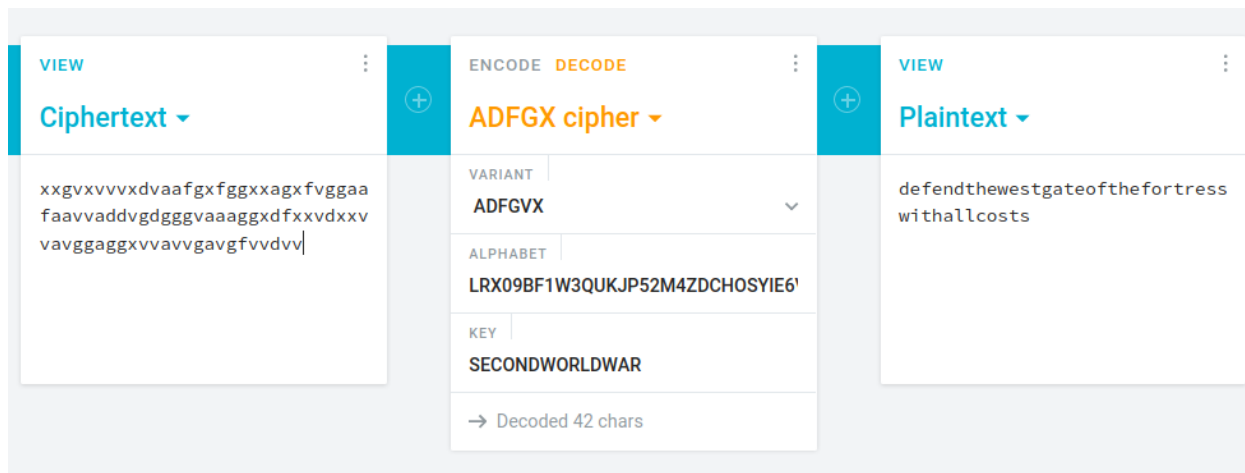
XXGVXVVXDVAAFGXFGGXXAGXFGGAAFAAVVADDVGDGGGVAAAGGXDFXXVDXXV  
VAVGGAGGXVAVVGAVGFVVDVV BEFORE THE OFFENSIVE THE ALLIES WERE  
VIRTUALLY BLIND TO GERMAN TROOP MOVEMENT. DURING THE LIBERATION OF  
FRANCE, THE EXTENSIVE NETWORK OF THE FRENCH RESISTANCE HAD PROVIDED  
VALUABLE INTELLIGENCE ABOUT GERMAN DISPOSITIONS. ONCE THEY REACHED  
THE GERMAN BORDER, THIS SOURCE DRIED UP. IN FRANCE, ORDERS HAD BEEN  
RELAYED WITHIN THE GERMAN ARMY USING RADIO MESSAGES ENCIPHERED BY  
THE ENIGMA MACHINE, AND THESE COULD BE PICKED UP AND DECRYPTED BY  
ALLIED CODE-BREAKERS HEADQUARTERED AT BLETCHLEY PARK, TO GIVE THE  
INTELLIGENCE KNOWN AS ULTRA. IN GERMANY SUCH ORDERS WERE TYPICALLY  
TRANSMITTED USING TELEPHONE AND TELEPRINTER, AND A SPECIAL RADIO  
SILENCE ORDER WAS IMPOSED ON ALL MATTERS CONCERNING THE UPCOMING  
OFFENSIVE. 42 THE MAJOR CRACKDOWN IN THE WEHRMACHT AFTER THE 20 JULY  
PLOT TO ASSASSINATE HITLER RESULTED IN MUCH TIGHTER SECURITY  
LRX09BF1W3QUKJP52M4ZDCHOSYIE6VG8NAT7 AND FEWER LEAKS. THE ATTACKS  
BY THE SIXTH PANZER ARMY'S INFANTRY UNITS IN THE NORTH FARED BADLY  
BECAUSE OF UNEXPECTEDLY FIERCE RESISTANCE BY THE U.S. 2ND AND 99TH  
INFANTRY DIVISIONS. KAMPFGRUPPE PEIPER, AT THE HEAD OF SEPP DIETRICH'S  
SIXTH PANZER ARMY, HAD BEEN DESIGNATED TO TAKE THE  
LOSHEIM-LOSHEIMERGRABEN ROAD, A KEY2: SECONDWORLDWAR ROUTE  
THROUGH THE LOSHEIM GAP, BUT IT WAS CLOSED BY TWO COLLAPSED  
OVERPASSES THAT GERMAN ENGINEERS FAILED TO REPAIR DURING THE FIRST  
DAY.

Pe parcursul mesajului se pot observa 3 stringuri 'interesante':

1.  
XXGVXVVXDVAAFGXFGGXXAGXFGGAAFAAVVADDVGDGGGVAAAGGXDFXXVDXXV  
VAVGGAGGXVAVVGAVGFVVDVV
2. LRX09BF1W3QUKJP52M4ZDCHOSYIE6VG8NAT7
3. KEY2: SECONDWORLDWAR

Cum exercițiul este în categoria **crypto**, putem presupune ca este vorba de încă un cifru. Al doilea string pare fi un alfabet sau o permutatie si al treilea cheia, ceea ce înseamnă că primul este **ciphertext**-ul. O particularitate usor observabila este prezenta a doar 6 litere în componenta ciphertext-ului: A, D, F, G, V si X. Dacă căutam aceste litere pe google alături de

cuvinte precum 'crypto' sau 'cipher' descoperim ca cifrul poate fi [ADFGVX Cipher](#). Putem folosi [Cryptii](#) pentru a recupera mesajul inițial:



Flag-ul este `ctf{sha256('defendthewes[redacted]sswithallcosts')}`.

Rezolvare în engleză: <https://blog.kuhi.to/unbreakable-romania-2-writeup#warplan>

## Secure-encryption (usor)

Concurs: UNbreakable 2021 #Individual

Descriere:

Decode the encryption and get the flag.  
Flag format CTF{sha256}

Rezolvare:

După ce ne conectăm la adresa furnizată în exercițiu, putem obține o mostră din această valoare criptată.

```
yakuhito@furry-catstation:~/ctf/unr21-ind$ nc 35.198.184.110 31412
What is the initial message of the encryption?
ENC= b'S#(HLcuz@hZ%0>IY*%k~T6J?(SY}pdP+C`ZL}z4a'
Value: ^C
yakuhito@furry-catstation:~/ctf/unr21-ind$
```

Odată ce știi că mesajul este criptat cu base85 / ascii85, realizarea unui script de rezolvare este ușoară:

```
from pwn import *
import base64

context.log_level = "CRITICAL"

r = remote("35.198.184.110", 31412)

def round():
    r.recvuntil(b"ENC= b")
    enc = r.recvuntil(b"")[::-1]
    dec = base64.b85decode(enc)
    print(dec)
    r.sendlineafter(b"Value: ", dec)

while True:
    round()
    line = r.recvline()
    if not line.startswith(b"What is the initial message"):
        flag = line.decode().strip().split(": ")[1]
        print(flag)
        break
```

Flag: CTF{d2e1793c6116d25fd592d[REDACTAT]06d5285ce6d1b157abdf10962}

## Rsa-quiz (medium)

Concurs: UNbreakable 2021 #Individual

Descriere:

We were trying to develop an AI-powered teacher, but it started giving quizzes to anyone who tries to connect to our server. It seems to classify humans as 'not sentient' and refuses to give us our flag. We really need that flag - can you please help us?

*Rezolvare:*

Până acum ar trebui să cunoaștem ce este criptarea RSA.

```
from Crypto.Util.number import inverse
from pwn import *
```

```
context.log_level = "CRITICAL"
r = remote("35.198.90.23", 30147)
```

```
# question 1
""""
```

```
_|||_ _|||_ _||
_|_|_|_|_|_|_|_|
_|_|_|_|_|_|_|_|
_|_|_|_|_|_|_|_|
_|_|_|_|_|_|_|_|
_|_|_|_|_|_|_|_|
```

Welcome! Today you are taking a quiz.

Here are some ground rules:

1. DON'T HAX - just answer the questions
2. When asked for a string / piece of text, just delete all non-alphanumeric characters and make sure your answer is in lowercase  
e.g. If the answer is 'John Cena', you should input 'johncena'.
4. Have fun! Just kidding... this is a quiz afterall

Let's start with something simple.

What does the S in RSA stand for? """"

```
r.recvuntil(b"??")
r.sendline(b"shamir")
```

```
# question 2
""""
```

```
If p is 19 and q is 3739, what is the value of n?
"""
p = 19
q = 3739
n = p * q
r.recvuntil(b"?.")
r.sendline(str(n).encode())

# question 3
"""
That was too simple! If n is 675663679375703 and q is 29523773, what is the value of p?
"""
n = 675663679375703
q = 29523773
p = n // q
r.recvuntil(b"?.")
r.sendline(str(p).encode())

# question 4
"""
Ok, I'll just give you something harder!
n=616571, e=3, plaintext=1337
Gimme the ciphertext:
"""
n = 616571
e = 3
plaintext = 1337
ciphertext = pow(plaintext, e, n)
r.recvuntil(b": ")
r.sendline(str(ciphertext).encode())

# question 5
"""
Maybe the numbers are too small...
e = 65537
p = 963760406398143099635821645271
q = 652843489670187712976171493587
Gimme the totient of n:
"""
e = 65537
```

```
p = 963760406398143099635821645271
q = 652843489670187712976171493587
phi = (p - 1) * (q - 1)
r.recvuntil(b": ")
r.sendline(str(phi).encode())

# question 6
"""
Oh, you know some basic math concepts... then give me d (same p, q, e):
"""
d = inverse(e, phi) # mod inv
r.recvuntil(b": ")
r.sendline(str(d).encode())

# question 7
"""
You do seem to exhibit some signs of intelligence. Decrypt
572595362828191547472857717126029502965119335350497403975777 using the same
values for e, p, and q (input a number):
"""
ciphertext = 572595362828191547472857717126029502965119335350497403975777
n = p * q
plaintext = pow(ciphertext, d, n)
r.recvuntil(b": ")
r.sendline(str(plaintext).encode())

# question 8
"""
Hmm.. Please encrypt the number 12345667890987654321 for me (same values for p, q, e):
"""
plaintext = 12345667890987654321
ciphertext = pow(plaintext, e, n)
r.recvuntil(b": ")
r.sendline(str(ciphertext).encode())

# question 9
"""
It appears that you might be sentient...
n =
152929646813683153154787333192209811374534931741180398509668504886770084711
```

```
52832453688156424015260891449686107937821564583408323587168077739041939832
44405517888812358757101255197456988935216581313608812764213989045789289145
428132470360886104251155582751423895206935681136093497324032887874358373932
62598817311
e = 65537
p =
117156630672524623341459077981169323946560224426262741399186848562274674772
60502860548284356112191762447814937304839893522375277179695353326622698517
979487
ciphertext =
929080756231565046072010381311510805340300704672918690741155645656737912019
955769470131211705776157512353159492753208306455977995853951482086611031565
68883014693664616195873778936141694426969384158471475412561910909609358186
64132317410588128108363045051396166801226371062061850988820299608255728934
3751590657
Tell me the plaintext (as a number):
""
n =
152929646813683153154787333192209811374534931741180398509668504886770084711
52832453688156424015260891449686107937821564583408323587168077739041939832
44405517888812358757101255197456988935216581313608812764213989045789289145
428132470360886104251155582751423895206935681136093497324032887874358373932
62598817311
e = 65537
p =
117156630672524623341459077981169323946560224426262741399186848562274674772
60502860548284356112191762447814937304839893522375277179695353326622698517
979487
ciphertext =
929080756231565046072010381311510805340300704672918690741155645656737912019
955769470131211705776157512353159492753208306455977995853951482086611031565
68883014693664616195873778936141694426969384158471475412561910909609358186
64132317410588128108363045051396166801226371062061850988820299608255728934
3751590657
q = n // p
phi = (p - 1) * (q - 1)
d = inverse(e, phi)
plaintext = pow(ciphertext, d, n)
r.recvuntil(b": ")
r.sendline(str(plaintext).encode())
```

```
# question 10
"""
Did you enjoy this quiz? (one word)
"""
r.recvuntil(b"word")
r.sendline(b"yes")

# get flag
r.recvuntil(b" Here's your reward:\n")
flag = r.recvline().decode().strip()

print(flag)
```

Flag: CTF{45d2f31123799facb31c46b757ed2cbd151ae8dd9798a9468c6f24ac20f91b90}

## Rgb (mediu)

*Concurs UNbreakable 2021 #Echipe*

*Descriere:*

Hue has messed up the pixels of this image and hid something **in** it.

The flag format: sha256.

Flag confirmation CTF{sha256}.

*Rezolvare:*



Analizând imaginea, putem observa că în colțul din dreapta jos se află o secvență de pixeli care au o formă regulată.

După o căutare rapidă despre rgb/hue și conexiunea criptografică "hue cipher" ne dăm seama că criptarea folosită pentru acei pixeli este hexahue.

Pentru a extrage secvența de pixeli putem folosi o pernă python sau putem potrivi manual steagul selectând un bloc de pixeli cu 2 coloane și 3 linii pentru fiecare literă sau număr.

## Crypto-twist (ușor)

Concurs UNbreakable 2021 #Echipe

Descriere:

1) Given the following sequence of bytes **in** hex format:

```
43 43 35 33 42 43 35 46 46 54 38 39 45 38 42 32 38 45 46 43 38 b7 b7 b7 7d 37 44 7b 39 35  
b7 b7 b7 35 32 44 33 36 42 b7 b7 b7 43 46 44 41 35 45 b7 b7 b7 34 41 43 43 42 44 45 30 37  
39 42 32 43 44 34 30 39 38 34 44 46 38 30 43 30 36 42 31 43 39
```

Can you get the flag?

2) Using the information from the first point, the best known symmetric encryption algorithm and the information below can you determine what the second flag is?

```
AF1E0A7857CF1D1403F8D359E8649C963DEF251A37EE91BDCC983A32917B177CF7253  
8C90533B0ACD9BFCC49F4703D2004991C10DDFBDB8FDE443C00246E288F3B86345392
```

Tag: 06D2BE58FB50064D56F49F9C725A1791

All flag formats: CTF{sha256}

Rezolvare:

Primul lucru pe care trebuie să-l facem este să transformăm aceste numere hexagonale în ASCII. Există un singur caracter ce nu este lizibil, 0xb7, pe care l-am înlocuit cu "X":

```
yakuhito@furry-catstation:~/ctf/unr21-tms$ python  
Python 3.6.9 (default, Jan 26 2021, 15:33:00)  
[GCC 8.4.0] on linux  
Type "help", "copyright", "credits" or "license" for more information.
```

```
>>> enc = "43 43 35 33 42 43 35 46 46 54 38 39 45 38 42 32 38 45 46 43 38 b7 b7 b7 7d 37  
44 7b 39 35 b7 b7 b7 35 32 44 33 36 42 b7 b7 b7 43 46 44 41 35 45 b7 b7 b7 34 41 43 43 42  
44 45 30 37 39 42 32 43 44 34 30 39 38 34 44 46 38 30 43 30 36 42 31 43 39"  
>>> enc = bytes.fromhex(enc.replace(" ", "").replace(b"\xb7", b"X").decode())  
>>> print(enc)  
CC53BC5FFT89E8B28EFC8XXX}7D{95XXX52D36BXXXC FDA5EXXX4ACCBDE079B2CD4  
0984DF80C06B1C9  
>>> exit()  
yakuhito@furry-catstation:~/ctf/unr21-tms$
```

Șirul de ieșire are 81 de caractere și conține numai caractere pe care le-am putut găsi într-un flag (plus câteva X-uri). Acest lucru înseamnă că flag-ul ar putea fi amestecat - permiteți-mi să îl aranjez într-o matrice 9x9:

```
CC53BC5FF  
T89E8B28E  
FC8XXX}7D  
{95XXX52D  
36BXXXC FD  
A5EXXX4AC  
CBDE079B2  
CD40984DF  
80C06B1C9
```

Vezi flag-ul? Dacă citim matricea ca pe o spirală, putem obține primul flag (jos, dreapta, sus, stânga, repetați). Spiral Cipher de pe dcode.fr ne poate ajuta să obținem primul flag fără a scrie un script.

Al doilea flag poate fi obținut prin decriptarea șirului dat cu AES-GCM. Putem împărți sha256 al primului flag în două părți de lungime egală: prima este cheia, iar a doua este iv. Iată rețeta CyberChef folosită în rezolvarea acestui exercițiu:

```
https://gchq.github.io/CyberChef/#recipe=AES_Decrypt(%7B'option':'Hex','string':'3ACC80C06  
B1C9F2CDDDEFF5CB35C8C96'%7D,%7B'option':'Hex','string':'5BD40984DBAF2782B8E98  
5BEDE0794C5'%7D,'GCM','Hex','Raw',%7B'option':'Hex','string':'06D2BE58FB50064D56F49  
F9C725A1791'%7D,%7B'option':'Hex','string':'"%7D)&input=QUYxRTBBNzg1N0NGMUQxND  
AzRjhEMzU5RTg2NDIDOTYzREVGMjUxQTM3RUU5MUJEQ0M5ODNBMzI5MTdCMTc3Q0Y  
3MjUzOEM5MDUzM0lwQUNEOUJGQ0M0OUY0NzAzRDIwMDQ5OTFDMTBEREZCREI4Rk  
RFNDQzQzAwMjQ2RTI4OEYzQjg2MzQ1Mzky
```

Flag 1: CTF{3ACC80C06B1C9F2CDD[REDACTAT]0984DBAF2782B8E985BEDE0794C5} Flag  
2: CTF{EFFD19F8DF40A7745469503[REDACTAT]452023A615EEC0A1B1BB5B73F37}

## Corrupt-decryption (greu)

Concurs UNbreakable 2021 #Echipe

Descriere:

Every time my friend sends me a message he encrypts it **for** security purposes, but this time I messed up the decrypter.

Can you repair the decryption program and decrypt the message?

Flag format CTF{sha256(message)}.

Rezolvare:

Executabilul este un program de decriptare care conține o eroare de memorie, respectiv una dintre funcții nu acceptă ieșirea funcției de decriptare anterioare pentru a continua decriptarea mesajului și duce la o eroare de segmentare (core dumped).

Scopul acestei provocări este de a inversa codul binar și de a extrage codul sursă, de a afla ce face codul și de a afla ce parte a codului produce această eroare.

Pentru a rezolva această problemă, ieșirea funcției anterioare trebuie formatată și trimisă următoarei funcții sub formă de șir de caractere, iar programul va decoda mesajul.

```
strcpy(ans, "XMESDXHHEDUXKFDYSDLGWDQPDXSQP");
```

## Enigma (mediu)

Concurs UNbreakable 2021 #Individual

Autor exercițiu: trollzorftw

Contribuitori rezolvare: Horia Niță, Valentina Galea

Descriere:

Can you crack this legendary code?

Rezolvare:

Aflați că primele 3 caractere reprezintă poziția rotorilor mașinii enigma, configurați o mașină enigma virtuală de bază și descifrați toate codurile de pe server pentru a obține steagul.

După ce ne conectăm la server, ni se solicită un mesaj de tipul `ZON|CDZQFNRRQJQ`

Răspuns: Ne dăm seama că primele trei caractere reprezintă poziția rotorilor mașinii enigma. De asemenea, ne dăm seama că următoarele 10 caractere reprezintă textul cifrat.

Apoi încercăm să mergem pe `https://www.dcode.fr/enigma-machine-cipher` și să încercăm să decodăm mesajul.

Am setat pozițiile inelului alfabetului la A, A, A, A (implicit), iar rotoarele la Z, O, N. După ce am setat și textul cifrat în interiorul căsuței "MESSAGE TO TYPE ON THE ENIGMA MACHINE", am scos și configurația plăcii de conectare (setările implicite).

Apăsăm pe decriptare și obținem `YZPAVAKUTW`. Dacă încercăm să trimitem acest lucru către server, obținem: `Prea lent. Nu sunt permise skiddies`.

Acest lucru înseamnă, că trebuie să creăm un script care să facă automat decodarea pentru noi.

După ce am căutăm puțin pe internet, am găsit o bibliotecă numită `pyenigma`.

`https://github.com/cedricbonhomme/pyEnigma`

Unde putem găsi un exemplu de script:

```
engine = enigma.Enigma(rotor.ROTOR_Reflector_A, rotor.ROTOR_I,
rotor.ROTOR_II, rotor.ROTOR_III, key="ABC",
plugs="AV BS CG DL FU HZ IN KM OW RX")
print(engine.enchiper("Hello World"))
We, however, want to use different settings for our enigma machine (the default ones).
engine = enigma.Enigma(rotor.ROTOR_Reflector_B, rotor.ROTOR_I,
rotor.ROTOR_II, rotor.ROTOR_III,
key=TBD)
# no plugs, reflector b is used.
```

Dacă testăm scriptul nostru de rezolvare, observăm că acesta funcționează pentru primele câteva (30-50) texte cifrate.

Cu toate acestea, după acestea, aparent primim un răspuns greșit.

Se pare că este vorba de un bug de pe server.

Din această cauză, am decis să deschidem conexiuni de fiecare dată când cea veche moare, până când obținem destule rezolvate pentru a obține steagul.

Ca atare, creăm următorul script de rezolvare:

```
from pwn import *
from pyenigma import *

# rot_pos = "IGV"
# cipher = "EOEYGNUOFQ"

# to_send = input("> ")
i = 0
while i <= 100:
    i = 0
    context.log_level = "debug"
    r = remote("34.159.190.67", 30974)
    while i <= 100:
        try:
            i += 1
            l = r.recvuntil(" ")[:-1].decode("utf-8")
            r.recvuntil(": ")
            rot_pos = l.split("|")[0]
            cipher = l.split("|")[1]
            engine = enigma.Enigma(rotor.ROTOR_Reflector_B,
                                   rotor.ROTOR_I, rotor.ROTOR_II, rotor.ROTOR_III,
                                   key=rot_pos)
            to_send = engine.encipher(cipher).strip()
            r.sendline(to_send)
        except EOFError:
            break
```

După câteva minute de la execuția scriptului, obținem următorul rezultat:

Congrats. You tamed enigma. Here's **your flag**:

```
ctf{5866e4563bd65bd6746ecb[REDACTAT]3852d5ea843427fb6dcdd7f1c}
```

## Prophet (hard)

Concurs UNbreakable 2021 #Individual

Autor exercițiu: Alexandru Bogatu ( Ephvuln )

Contribuitori rezolvare: Horia Niță, Valentina Galea

Descriere:

The prophecy may take time.

Rezolvare:

După ce ne conectăm la server, încercăm să ne jucăm cu câteva comenzi.

Observăm că în meniu avem 2 opțiuni.

```
Welcome to my priv
1. Receive message
2. Send message
```

Selectăm 2, iar apoi ni se cere să introducem un mesaj.

Dacă încercăm să apăsăm pur și simplu enter, fără niciun mesaj, primim această eroare:

```
Traceback (most recent call last):
  File "/home/ctf/server.py", line 70, in <module>
    menu()
  File "/home/ctf/server.py", line 61, in menu
    if is_padding_ok(cipher):
  File "/home/ctf/server.py", line 44, in is_padding_ok
    return _decrypt(data) is True
  File "/home/ctf/server.py", line 35, in _decrypt
    cipher = AES.new(_key, AES.MODE_CBC, iv)
  File "/usr/local/lib/python3.9/dist-packages/Crypto/Cipher/AES.py", line 232, in new
    return _create_cipher(sys.modules[__name__], key, mode, *args, **kwargs)
  File "/usr/local/lib/python3.9/dist-packages/Crypto/Cipher/_init_.py", line 79, in
    _create_cipher
    return modes[mode](factory, **kwargs)
  File "/usr/local/lib/python3.9/dist-packages/Crypto/Cipher/_mode_cbc.py", line 287, in
    _create_cbc_cipher
    raise ValueError("Incorrect IV length (it must be %d bytes long)" %
```

```
ValueError: Incorrect IV length (it must be 16 bytes long)
```

Din această eroare, putem deduce câteva lucruri

1. Serverul utilizează o criptare AES pentru textele cifrate.
2. Serverul utilizează AES în modul CBC. ( cipher = AES.new(\_key, AES.MODE\_CBC, iv) )
3. IV este, din anumite motive, o intrare? (mai puțin important)
4. Dacă selectăm 1, primim un mesaj de la Bob, cu textul cifrat în care flag-ul este  
CxVj8/82j8v4Yo8nrOV0Hv1NOIWEIvyLz6b3ex3g/SdWNEEcsEFo/FfC0efQ58Cb8pSznk62aVz  
bEA9kr/XGYY+hsAxbNvvvgfyyI27NH1PmAuNi3ZIKaYCNmRV96Nqs

Apoi putem încerca să trimitem acest șir de caractere ca mesaj și obținem următorul rezultat:

Opțiunea: 2

```
Msg:CxVj8/82j8v4Yo8nrOV0Hv1NOIWEIvyLz6b3ex3g/SdWNEEcsEFo/FfC0efQ58Cb8pSznk6  
2aVzbEA9kr/XGYY+hsAxbNvvvgfyyI27NH1PmAuNi3ZIKaYCNmRV96Nqs Ack.
```

Acum putem încerca să mai trimitem câteva lucruri către server, cum ar fi vechiul mesaj criptat pe care Bob l-a trimis.

Dacă facem asta, obținem următorul lucru:

```
Msg:rZjY2yoyR6P208xSFGVnMiUme0LURDpPz+OILig3ERsj9KzegVW/IOVIC+QofhUPGAly4  
LkblsshKaCNNTL7MzUR32wSFYUsAIMJ5rUmd3Ss7Ycf1q+24nzgl9JeSEy0  
Padding seems invalid
```

Mesaj de umplură invalid și CBC AES. Acest lucru necesită un atac de tip "padding oracle". Căutând pe Internet putem găsi următorul script, ce este actualizat pentru Python3

```
https://github.com/stephenbradshaw/python-paddingoracle/tree/0af070cba117eec555a4157095d9474a603df616
```

În acest moment scriptul nostru de rezolvare arată așa:

```
from paddingoracle import BadPaddingException, PaddingOracle
from pwn import *

r = remote("34.159.235.104", 31907)
r.recvuntil("Option: ")
r.sendline("1")
work_on = r.recvline().decode("utf-8").split(" ")[1]

class PadBuster(PaddingOracle):
    def __init__(self, **kwargs):
        super(PadBuster, self).__init__(**kwargs)

    def oracle(self, data, **kwargs):
        rl = r.recvuntil("Option: ")
        print(rl)
        r.sendline("2")
        rl = r.recvuntil("Msg:")
        print(rl)
        print(b64e(data))
        r.sendline(b64e(data))
        inv = r.recvline()
        print(inv)
        if inv == b'Padding seems invalid\r\n':
            raise BadPaddingException
        return

        # work_on =
        hexlify(base64.b64decode(work_on)).decode("utf-8")

if __name__ == '__main__':
    print(work_on)
```



```
data = b64d(work_on)
padbuster = PadBuster()
res = padbuster.decrypt(data, block_size=16,
iv=bytearray(16))
print(b64e(res))
```

După ce a rulat pentru o perioadă destul de lungă de timp, obținem

```
0+UOV7rmI8MGCSCSMn9Mv5nUNURns0NDFjYzAxMmJjYjE4MDJhMzAwYzI3NjcyNzFhOTI
5NzYxM2U1Y2QwMzc0OTY0NzlkYTdkMjRjRiMjI1MWRkNGE2fQsLCwsLCwsLCwsLCwsL
```

Este un șir de caractere base64, pe care îl putem decoda folosind

<https://www.base64decode.org/> sau CyberChef

Astfel, obținem steagul nostru înconjurat de o grămadă de zgomot.

```
Óâ.W°æ#Ã.
#ôËÛ.CTF{441cc012bcb1802a300c2767271a[REDACTAT]479da7d24b2251dd4a6}.....
```

## Communication (ușor)

Concurs UNbreakable 2021 #Echipe

Autor exercițiu: Daniel Popovici ( betaflash )

Contribuitori rezolvare: Andrei Albișoru, Teodor Duțu, Adina Smeu, Valentina Galea

Descriere:

Analyze this pcap and extract the password.

Rezolvare:

Am inspectat captura cu Wireshark și am observat că aceasta conținea handshake-uri cu 4 căi pentru conectarea la un punct de acces.

Am folosit apoi aircrack-ng cu lista de cuvinte rockyou.txt pentru a sparge parola din captură (<https://securitytutorials.co.uk/how-to-capture-crack-wpa-wpa2-wireless-passwords/>):

```
adina@asm:communication$ aircrack-ng -w ../../../../rockyou.txt authentication.pcap
Reading packets, please wait...
Opening authentication.pcap
Read 427 packets.
```

```
# BSSID ESSID Encryption
1 DC:A6:32:AC:DA:6B test-raspi WPA (1 handshake) Choosing first network as target.
Reading packets, please wait...
Opening authentication.pcap
Read 427 packets.
1 potential targets
Aircrack-ng 1.6
[00:00:01] 7750/10303727 keys tested (11703.62 k/s)
Time left: 14 minutes, 39 seconds 0.08%
KEY FOUND! [ firefighter ]
Master Key : 50 A1 25 3F 44 A5 82 43 BA C6 79 FE C5 E5 4F E8 F9 C1 21 65 23 C2 03 16
E6 A2 C4 F5 83 E2 EA 95
Transient Key : BE E0 00 3D 5C 58 77 33 9C 9A EB 39 0E 15 E4 5D 87 72 C8 A8 93 C1 59
30 59 51 DE 60 53 A6 CC BD
00 5F FE 3B 40 46 D5 97 BD 63 BF BE 4B 0B D8 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
EAPOL HMAC : 04 21 03 CA AC 25 01 F6 8A 3C DB 03 BB 4A 60 16
```

Astfel obținem parola SHA256 și putem obține flag-ul.

## Elgamal-quiz (ușor)

Concurs UNbreakable 2021 #Echipe

Autor exercițiu: yakuhito

Contribuitori rezolvare: Andrei Albișoru, Teodor Duțu, Adina Smeu, Valentina Galea

Descriere:

Time for a quiz!

Rezolvare:

A trebuit să creăm un script python care să interacționeze cu un server și să răspundă la câteva întrebări legate de modul în care funcționează algoritmul elgamal.

După ce am răspuns la toate întrebările legate de modul în care sunt alese unele valori ale algoritmului și cum sunt calculate altele, decriptăm un șir de caractere, pe care îl trimitem serverului și primim în schimb flag-ul.

Am început implementarea din acest github care a fost de la un alt CTF. În cazul nostru p din script a fost q. Pentru a înțelege algoritmul am folosit wikipedia. Scriptul nostru arăta astfel:

Wiki: [https://en.wikipedia.org/wiki/ElGamal\\_encryption](https://en.wikipedia.org/wiki/ElGamal_encryption)

Github:

<https://github.com/hellok/CTF/blob/ccfa8d2a8de46f156e9cd490bd72deaf25ba8688/crypto/problem/elgamal.py>

```
from pwn import *
from gmpy2 import *

def elgamal_gen_x_and_h(q, g):
    x = random.randrange(1, q-1)
    h = pow(g, x, q)
    return (h, x)

io = remote("34.159.190.67", 30289)

msg = io.recvuntil(b'? ')

msg = msg.split(b"\n")[-1]
```

```
msg = msg.split(b' ')

q = int(msg[1][2:].decode('utf-8'))

g = int(msg[3][2:7].decode('utf-8'))

(h, x) = elgama1_gen_x_and_h(q, g)

io.sendline(str(x))
msg = io.recvuntil(b"h: ")

io.sendline(str(h))

msg = io.recvuntil(b": ")

m = int("0x6869", 16) # this is hi

io.sendline(str(m))

msg = io.recvuntil(b"? ")

y = random.randrange(1, q)

io.sendline(str(y))

msg = io.recvuntil(b"s: ")

s = pow(h, y, q)

io.sendline(str(s))

msg = io.recvuntil(b"c1: ")

c1 = pow(g, y, q)

io.sendline(str(c1))

msg = io.recvuntil(b"c2: ")
```

```
c2 = s * m % q

io.sendline(str(c2))

msg = io.recvline()

msg = io.recvline()

msg = msg.split(b"=")[1].lstrip(b"

").rstrip(b")\n").split(b", ") public =

[int(el.decode("utf-8")) for el in msg]

q = public[0]
g = public[1]
h = public[2]

msg = io.recvline()

msg = msg.split(b"=")[1].lstrip(b"

").rstrip(b")\n").split(b", ") cipher =

[int(el.decode("utf-8")) for el in msg]

c1 = cipher[0]
c2 = cipher[1]

msg = io.recvline()

msg = msg.split(b": ")[1].rstrip(b"\n")

x = int(msg.decode("utf-8"))

s = pow(c1, x, q)

msg = io.recvuntil(b": ")
```

```
io.sendline(str(s))

msg = io.recvuntil(b"s_inv: ")

s_inv = pow(s, -1, q)

io.sendline(str(s_inv))

msg = io.recvuntil(b": ")

m = bytes.fromhex(hex(c2 * s_inv %
q)[2:]).decode('ASCII') io.sendline(m)

msg = io.recvline()

print(msg)

io.close()
```

## Entangle (greu)

*Concurs UNbreakable 2021 #Echipe*

*Autor exercițiu: Alexandru Bogatu ( Ephvuln )*

*Contribuitori rezolvare: Andrei Albișoru, Teodor Duțu, Adina Smeu, Valentina Galea*

Descriere:

Bob wanted to make his own CBC but he might have mixed something up.

Rezolvare:

Serverul criptează prefixul + mesajul + steagul cu AES în modul ECB, utilizând o dimensiune a blocului aleasă aleatoriu de 16 sau 32.

Apoi, la blocurile criptate se adaugă un IV, iar fiecare dintre acestea se compară cu blocul anterior (primul bloc se compară cu IV).

Cea de-a doua "criptare" este irelevantă și se anulează prin xorarea blocurilor în același mod, dar în sens invers.

Prima criptare este vulnerabilă deoarece 2 blocuri identice de text în clar produc același bloc criptat.

Acest comportament permite identificarea lungimii prefixului.

Din acest punct, steagul poate fi divulgat octet cu octet, continuând să se folosească un atac de tip "chosen plaintext".

Primii pași care se fac la primirea oricărui răspuns de la server sunt împărțirea acestuia în blocuri și xorarea a două blocuri adiacente, începând de la capăt, pentru a anula xorarea efectuată de server la sfârșit.

Acest lucru este posibil deoarece serverul xor-ează blocurile după cum urmează:

Fie  $b[i]$  cel de-al  $i$ -lea bloc neexortat, iar  $b'[i]$  cel de-al  $i$ -lea bloc exortat.

Nu există  $b'[0]$  deoarece nu există niciun bloc înainte de primul.

$$b'[i] = b[i] \oplus b'[i - 1] \Rightarrow b[i] = b'[i] \oplus b'[i - 1]$$

Această primă etapă de procesare este realizată de funcția `get_blocks`:

```
def get_blocks(p):  
    enc = b64decode(p.recvline().rstrip())[4:]  
    blocks = split_blocks(enc)  
    for i in reversed(range(1, len(blocks))):  
        blocks[i] = xor(blocks[i - 1], blocks[j])  
    return blocks
```

16 au fost alese, spre deosebire de 32, din cauza calculului mai rapid pe care îl aduce.

După cum se menționează în rezumat, 2 blocuri identice de text în clar produc blocuri identice de text cifrat atunci când se utilizează modul ECB.

Pentru a deduce lungimea prefixului, trebuie să trimitem serverului un mesaj suficient de lung pentru a umple ultimul bloc al prefixului și pentru a umple 2 blocuri suplimentare.

Acest mesaj este doar un șir de "a".

În acest fel, cele două blocuri suplimentare vor fi identice în textul cifrat.

Astfel, decalajul cu care am "căptușit" blocul final al prefixului este  $\text{len}(\text{input}) - 2 * \text{BLOCK\_SIZE}$ . În cazul nostru, acest decalaj este 6.

Se determină prin trimiterea de intrări de diferite lungimi compuse din același caracter ("a") până când 2 blocuri sunt identice:

```
for j in reversed(range(1, len(blocks))):  
    if blocks[j] == blocks[j - 1]:  
        offset = i - (j - 1) * BLOCK_SIZE  
        found_block = True
```

Cunoscând acest decalaj, putem trimite sarcini utile care sunt cu 1 octet mai mici decât un bloc complet.

Blocul de text cifrat care corespunde ultimului bloc din textul în clar (prefix + intrare + steag) reprezintă criptarea unui bloc al cărui ultim octet este primul octet al steagului.

Prin trimiterea unei încărcături utile cu 2 octeți mai mică decât un bloc complet, 2 octeți ai steagului vor fi incluși în ultima criptare.

În acest fel, putem scăpa steagul octet cu octet, astfel:

```
flag = 'CTF{'  
for _ in range(64):  
    payload = b'a' * (offset + 5 * BLOCK_SIZE -  
        len(flag) - 1) p.sendline(payload)  
    blocks = get_blocks(p)  
    correct_block = blocks[6]  
  
    for i in HEX_LETTERS:  
        new_payload = payload + (flag + i).encode()  
        p.sendline(new_payload)  
        new_blocks = get_blocks(p)  
        if new_blocks[6] == correct_block:  
            flag += i  
            break
```

Scriptul rezultat poate fi văzut mai jos:

```
from pwn import *  
from base64 import b64decode  
from math import ceil
```



```
BLOCK_SIZE = 16 # shorter running time
HEX_LETTERS = 'abcdef0123456789'

context.log_level = 'error'
def split_blocks(x):
    n = ceil(len(x) / BLOCK_SIZE)
    return [x[BLOCK_SIZE*i : BLOCK_SIZE*(i+1)] for i in range(n)]

def xor(x, y):
    return bytes(a ^ b for (a, b) in zip(x, y))

def get_blocks(p):
    enc = b64decode(p.recvline().rstrip()[4:])
    blocks = split_blocks(enc)
    for i in reversed(range(1, len(blocks))):
        blocks[i] = xor(blocks[i - 1], blocks[i])
    return blocks

# https://zachgrace.com/posts/attacking-ecb/
found_block = False
while not found_block:
    p = remote('34.159.190.67', 30209)
    p.recvline()

    for i in range(2 * BLOCK_SIZE, 3 * BLOCK_SIZE):
        p.sendline(b'a' * i)
        blocks = get_blocks(p)

        for j in reversed(range(1, len(blocks))):
            if blocks[j] == blocks[j - 1]:
                offset = i - (j - 1) * BLOCK_SIZE
                found_block = True

    if found_block:
        break
```

```
flag = 'CTF{'
for _ in range(64):
    payload = b'a' * (offset + 5 * BLOCK_SIZE -
len(flag) - 1) p.sendline(payload)
    blocks = get_blocks(p)
    correct_block = blocks[6]

    for i in HEX_LETTERS:
        new_payload = payload + (flag + i).encode()
        p.sendline(new_payload)
        new_blocks = get_blocks(p)
        if new_blocks[6] == correct_block:
            flag += i
            break

print(flag + '}' )
```

## Surfing (mediu)

Concurs UNbreakable 2021 #Echipe

Autor exercițiu: Daniel Popovici ( betaflash )

Contribuitori rezolvare: Andrei Albișoru, Teodor Duțu, Adina Smeu, Valentina Galea

Descriere:

Someone leaked the flag over the internet.

Rezolvare:

Am observat că keys.log conținea mai multe chei SSL/TLS și că captura conținea numai pachete criptate. Am căutat online cum să import acele chei în wireshark și am găsit un articol cu aceste instrucțiuni:

Accesați Editare > Preferințe > Protocoale > TLS.  
Pentru numele fișierului jurnal (Pre)-Master-Secret, selectați Browse  
(Răsfoiește) și localizați fișierul jurnal SSL pe care l-ați creat.

Apoi am căutat în detaliile pachetului șirul CTF{, dar am găsit un indicator incomplet:

```
/method/links.getStats?urls=https://bit-sentinel.com/search/?flag1=CTF{4fa2  
7628dd92107/&format=json&callback=jQuery1102004610531156523345_163913247882  
0&_=1639132478821
```

Acum putem căuta și după flag2 și flag3, astfel găsim părțile lipsă:

```
/?id=https://bit-sentinel.com/search/?flag2=75c76263c0d6bef0f86b80e/&callba  
ck=jQuery110207471494685464508_1639132518533&_=1639132518534  
Form item: "href" =  
"https://bit-sentinel.com/search/?flag3=3fef78c072879d639e34ba6734}/"
```

## Contribuitori

- Mihai Dancaescu (yakuhto)
- Horia Niță
- Popovici Daniel (betaflash)
- CertSIGN
- Andrei Albișoru
- Teodor Duțu
- Adina Smeu