**Cryptography Fundamentals and Pentesting Basics**

Caesar Cipher (2p):   
● Task: decrypt given cyphertext: “Hvs Eiwqy Pfckb Tcl Xiadg Cjsf Hvs Zonm Rcu.”

● Challenge: Perform Frequency Analysis or brute-force attack to decrypt a ciphertext. Provide python code solution with GitHub.   
● Discussion: Why is Caesar cipher insecure? Where might legacy systems still use similar encryption?

**Code :**

def decrypt\_caesar\_cipher(text, shift):

result = ""

for i in range(len(text)):

char = text[i]

if char.isupper():

result += chr((ord(char) - shift - 65) % 26 + 65)

elif char.islower():

result += chr((ord(char) - shift - 97) % 26 + 97)

else:

result += char

return result

def brute\_force\_caesar\_cipher(cipher\_text):

for shift in range(1, 26):

decrypted\_text = decrypt\_caesar\_cipher(cipher\_text, shift)

print(f"Shift {shift}: {decrypted\_text}")

cipher\_text = "Hvs Eiwqy Pfckb Tcl Xiadg Cjsf Hvs Zonm Rcu."

brute\_force\_caesar\_cipher(cipher\_text)

**Output:**  
Shift 1: Gur Dhvpx Oebja Sbk Whzcf Bire Gur Ynml Qbt.

Shift 2: Ftq Cguow Ndaiz Raj Vgybe Ahqd Ftq Xmlk Pas.

Shift 3: Esp Bftnv Mczhy Qzi Ufxad Zgpc Esp Wlkj Ozr.

Shift 4: Dro Aesmu Lbygx Pyh Tewzc Yfob Dro Vkji Nyq.

Shift 5: Cqn Zdrlt Kaxfw Oxg Sdvyb Xena Cqn Ujih Mxp.

Shift 6: Bpm Ycqks Jzwev Nwf Rcuxa Wdmz Bpm Tihg Lwo.

Shift 7: Aol Xbpjr Iyvdu Mve Qbtwz Vcly Aol Shgf Kvn.

Shift 8: Znk Waoiq Hxuct Lud Pasvy Ubkx Znk Rgfe Jum.

Shift 9: Ymj Vznhp Gwtbs Ktc Ozrux Tajw Ymj Qfed Itl.

Shift 10: Xli Uymgo Fvsar Jsb Nyqtw Sziv Xli Pedc Hsk.

Shift 11: Wkh Txlfn Eurzq Ira Mxpsv Ryhu Wkh Odcb Grj.

Shift 12: Vjg Swkem Dtqyp Hqz Lworu Qxgt Vjg Ncba Fqi.

Shift 13: Uif Rvjdl Cspxo Gpy Kvnqt Pwfs Uif Mbaz Eph.

**Shift 14: The Quick Brown Fox Jumps Over The Lazy Dog.**

Shift 15: Sgd Pthbj Aqnvm Enw Itlor Nudq Sgd Kzyx Cnf.

Shift 16: Rfc Osgai Zpmul Dmv Hsknq Mtcp Rfc Jyxw Bme.

Shift 17: Qeb Nrfzh Yoltk Clu Grjmp Lsbo Qeb Ixwv Ald.

Shift 18: Pda Mqeyg Xnksj Bkt Fqilo Kran Pda Hwvu Zkc.

Shift 19: Ocz Lpdxf Wmjri Ajs Ephkn Jqzm Ocz Gvut Yjb.

Shift 20: Nby Kocwe Vliqh Zir Dogjm Ipyl Nby Futs Xia.

Shift 21: Max Jnbvd Ukhpg Yhq Cnfil Hoxk Max Etsr Whz.

Shift 22: Lzw Imauc Tjgof Xgp Bmehk Gnwj Lzw Dsrq Vgy.

Shift 23: Kyv Hlztb Sifne Wfo Aldgj Fmvi Kyv Crqp Ufx.

Shift 24: Jxu Gkysa Rhemd Ven Zkcfi Eluh Jxu Bqpo Tew.

Shift 25: Iwt Fjxrz Qgdlc Udm Yjbeh Dktg Iwt Apon Sdv.

**Decryption:** **The Quick Brown Fox Jumps Over The Lazy Dog.  
  
  
  
Why is Caesar cipher insecure?  
  
The Caesar cipher, while historically significant, is considered insecure due to its simplicity.**

1. **Limited Key Space:** The Caesar cipher only has 25 possible shifts (excluding the trivial shift of 0). This makes it highly susceptible to brute-force attacks, where an attacker can try all possible shifts to decrypt the message.
2. **Pattern Recognition:** Since the Caesar cipher simply shifts letters, the frequency of letters in the ciphertext will closely match the frequency in the plaintext. This makes it vulnerable to frequency analysis, where an attacker analyzes the frequency of letters or groups of letters to deduce the key.
3. **Simple Reversibility:** Encrypting and decrypting using the Caesar cipher involves simple modular arithmetic, making it easy to reverse.

**Where might legacy systems still use similar encryption?**

Despite its weaknesses, variations of the Caesar cipher or similar simple substitution ciphers might still be used in certain legacy systems, often for non-critical purposes where high security is not a primary concern. Examples include:

1. **Educational Tools:** Used in educational settings to teach basic concepts of encryption and decryption.
2. **Simple Games and Puzzles:** Often used in escape rooms, treasure hunts, or simple puzzles where the focus is on the fun challenge rather than security.
3. **Obfuscation:** In some legacy systems, basic ciphers might be used to obfuscate information rather than secure it. For example, simple text-based games or old software might use these methods to hide clues or instructions from users until the correct condition is met.
4. **Historical Systems:** Some very old legacy systems that have not been updated might still use these methods for compatibility reasons.

2. XOR Encryption/Decryption (3p):

Step 1: Caesar Cipher Challenge   
● Ciphertext: mznxpz   
● Challenge: Perform a brute-force or frequency analysis attack to decrypt the Caesar-encrypted text.   
● Clue: The ciphertext is an encrypted anagram of the passphrase.

Ciphertext: mznxpz

Clue: The ciphertext is an encrypted anagram of the passphrase.

Ciphertext: mznxpz

Decryption: rescue

Step 2: Solve the Anagram

Hint: The final passphrase is a fundamental concept in cryptography.

Anagram: rescue

Passphrase: SECURE (https://www.anagrammer.com/anagrams-of/rescue)

Step 3: XOR Decryption

Instructions:

1. Use the recovered passphrase from Step 2 to XOR-decrypt the message (first

you must convert base64 then decrypt).

Pythin script:

"""

import base64

# Given Base64 ciphertext

ciphertext\_base64 = "Jw0KBlIMAEUXHRdFKyoxVRENEgkPEBwCFkQ="

# Decode Base64 to bytes

ciphertext = base64.b64decode(ciphertext\_base64)

# Passphrase and preparation

passphrase = "secure"

key = (passphrase \* (len(ciphertext) // len(passphrase) + 1))[:len(ciphertext)]

key\_bytes = key.encode()

# XOR decryption

plaintext = bytes([c ^ k for c, k in zip(ciphertext, key\_bytes)])

# Result

print("Decrypted plaintext:", plaintext.decode())

"""

Answer:

Decrypted plaintext: This is the XOR challenge!