**KRYPTON**

## TEAM MEMBERS:

* Prajakta Shinde - KRYPTON
* Saniya Shinde - NATAS
* Riya Rasal - NATAS
* Shraddha Khade - NATAS
* Kumkum Yadav - LEVIATHAN

**Krypton Level 0 → Level 1**

bash

echo S1JZUFRPTklTR1JFQVQ= | base64 -d

**Krypton Level 1 → Level 2**

bash

cat /krypton/krypton1/krypton2

*# Python script for Caesar shift analysis*

for i in range(26):

shifted = ""

for char in ciphertext:

if char.isalpha():

ascii\_offset = ord('A') if char.isupper() else ord('a')

shifted += chr((ord(char) - ascii\_offset + i) % 26 + ascii\_offset)

else:

shifted += char

print(f"Shift {i}: {shifted}")

**Krypton Level 2 → Level 3**

bash

mktemp -d

cd /tmp/tmpXXXX

ln -s /krypton/krypton2/keyfile.dat

chmod 777 .

echo "ABCDEFGHIJKLMNOPQRSTUVWXYZ" > testfile

/krypton/krypton2/encrypt testfile

cat /krypton/krypton2/krypton3 | tr 'A-Z' 'M-ZA-L'

**Krypton Level 3 → Level 4**

bash

cat /krypton/krypton3/krypton4

cat /krypton/krypton3/found1

cat /krypton/krypton3/found2

cat /krypton/krypton3/found3

*# Frequency analysis commands*

cat /krypton/krypton3/found\* | grep -o . | sort | uniq -c | sort -nr

*# Substitution decryption (example using tr)*

cat /krypton/krypton3/krypton4 | tr 'ABCDEFGHIJKLMNOPQRSTUVWXYZ' 'DECRYPTION\_KEY\_HERE'

**Krypton Level 4 → Level 5**

bash

cat /krypton/krypton4/krypton5

cat /krypton/krypton4/found\*

*# Python script for column-based Vigenère analysis*

python3 -c '

ciphertext = "CONTENT\_OF\_KRYPTON5\_FILE"

key\_length = 6

# Split into columns

columns = ["" for \_ in range(key\_length)]

for i, char in enumerate(ciphertext):

if char.isalpha():

columns[i % key\_length] += char

# Analyze each column

key = ""

for column in columns:

# Frequency analysis for each column

frequencies = [0] \* 26

for char in column:

frequencies[ord(char.upper()) - ord("A")] += 1

# Determine shift that makes E most common

most\_common = frequencies.index(max(frequencies))

shift = (most\_common - (ord("E") - ord("A"))) % 26

key += chr(shift + ord("A"))

print("Vigenère key:", key)

'

**Krypton Level 5 → Level 6**

bash

cat /krypton/krypton5/krypton6

cat /krypton/krypton5/found\*

*# Python script for Kasiski examination and IoC analysis*

python3 -c '

ciphertext = "CONTENT\_OF\_KRYPTON6\_FILE"

# Find repeating sequences

def find\_repeats(text, min\_length=3):

repeats = {}

for i in range(len(text)-min\_length+1):

sequence = text[i:i+min\_length]

if sequence in text[i+1:]:

if sequence not in repeats:

repeats[sequence] = []

pos = i

while True:

pos = text.find(sequence, pos+1)

if pos == -1:

break

repeats[sequence].append(pos)

return repeats

# Calculate Index of Coincidence

def calculate\_ioc(text):

freq = [0] \* 26

for char in text:

if char.isalpha():

freq[ord(char.upper()) - ord("A")] += 1

n = sum(freq)

ioc = 0

if n > 1:

for count in freq:

ioc += count \* (count - 1)

ioc /= n \* (n - 1)

return ioc

# Test different key lengths

for key\_length in range(1, 20):

columns = ["" for \_ in range(key\_length)]

for i, char in enumerate(ciphertext):

if char.isalpha():

columns[i % key\_length] += char

avg\_ioc = sum(calculate\_ioc(col) for col in columns) / key\_length

print(f"Key length {key\_length}: Average IoC = {avg\_ioc:.4f}")

'

**Krypton Level 6 → Level 7**

bash

hexdump -C /krypton/krypton6/plain1 > plain1.hex

hexdump -C /krypton/krypton6/cipher1 > cipher1.hex

/krypton/krypton6/encrypt6 testinput

*# Python script for XOR operations*

python3 -c '

def xor\_bytes(a, b):

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

with open("/krypton/krypton6/plain1", "rb") as f:

plaintext = f.read()

with open("/krypton/krypton6/cipher1", "rb") as f:

ciphertext = f.read()

# Extract keystream

keystream = xor\_bytes(plaintext, ciphertext)

# Find keystream length through pattern analysis

keystream\_length = 0

for i in range(1, len(keystream)):

if keystream[0:i] == keystream[i:i\*2]:

keystream\_length = i

break

print(f"Keystream length: {keystream\_length}")

# Use keystream to decrypt password

with open("/krypton/krypton6/krypton7", "rb") as f:

encrypted\_password = f.read()

password = ""

keystream = keystream[:keystream\_length] # Just use one period

for i, byte in enumerate(encrypted\_password):

password += chr(byte ^ keystream[i % keystream\_length])

print(f"Password: {password}")

'