Krypton

Krypton0

Precisamos decodificar a mensagem `S1JZUFRPTklTR1JFQVQ=` para o próximo nível.

É dito que ela está em **base64**. Para resolver isso, é só usar o comando `base64`.

```
echo "S1JZUFRPTklTR1JFQVQ=" | base64 -d
```

Assim, obtemos a senha de Krypton1.

Resposta: `KRYPTONISGREAT`.

Krypton1

É preciso decodificar a mensagem `YRIRY GJB CNFFJBEQ EBGGRA`, encriptada por **rotação simples**.

Rotação simples pode significar alguma versão da Cifra de César, provavelmente ROT13. Para isso, podemos usar sites como dcode ou criar um programa em python que automatize isso:

```
def rot(char, shift):
    return chr((ord(char) - ord('A') + shift)%26 + ord('A'))

def decode(ciphertext, shift):
    msg = ''
    for c in ciphertext:
        msg += rot(c, shift) if c.isalpha() else c
    return msg

ciphertext = 'YRIRY GJB CNFFJBEQ EBGGRA'
shift = 13
flag = decode(ciphertext, shift)
print(flag)
```

A resposta está na cifra decodificada por ROT13: `LEVEL TWO PASSWORD ROTTEN`.

Resposta: `ROTTEN`.

Krypton2

Temos no arquivo `krypton3` a mensagem `OMQEMDUEQMEK` que está criptografada em **Cifra de César**.

Há duas formas de resolver esse nível.

Podemos criar uma pasta em `/tmp/` e linkar a keyfile nele para usarmos o executável `encrypt`. Dessa forma, podemos encriptar um arquivo com `a` para descobrir a rotação usada.

Com isso, o arquivo encriptado terá a letra `m`, indicando que foi usado ROT12 para encriptação e deverá ser usado ROT14 para decriptação, obtendo a *flag*.

Outro modo, muito mais rápido, é usar o mesmo método do nível anterior, fazendo um *brute-force* das 26 rotações possíveis de uma Cifra de César.

```
def brute_force(ciphertext):
    for shift in range(26):
        print(decode(ciphertext, shift))

brute_force('OMQEMDUEQMEK')
```

A que tiver a frase mais legível é a *flag*.

Resposta: `CAESARISEASY`.

Krypton3

Para esse nível, temos no arquivo `krypton4` a mensagem `KSVVW BGSJD SVSIS VXBMN YQUUK BNWCU ANMJS`, cifrada por alguma **cifra de substituição simples**. Nesse momento, usar *brute-force* não é mais uma opção.

Para resolver isso, podemos pegar o texto em `found1` e aplicar uma análise de frequência em suas letras.

Uma ferramenta online extremamente eficiente está localizada em guabala.de. Utilizando sua análise de frequência, obtemos o mapeamento:

```
abcdefghijklmnopqrstuvwxyz     This clear text ...
qazwsxedcrfvtgbyhnujmikolp     ... maps to this cipher text
```

Podemos, assim, decriptar a mensagem com o site dcode utilizando o alfabeto de susbtituição acima. O resultado é o texto `WELLD ONETH ELEVE LFOUR PASSW ORDIS BRUTE`.

Resposta: `BRUTE`.

Krypton4

Nesse nível, há a mensagem `HCIKV RJOX` em `krypton5`, codificada pela **Cifra de Vigenère**.

Primeiro, precisamos descobrir a *key* da cifra. Para isso usaremos o texto em `found1` e a ferramenta em dcode. Assim, encontramos a *key* `FREKEY`.

Dessa forma, utilizando novamente o dcode com a *key*, obtemos a *flag*.

Resposta: `CLEARTEXT`.

Krypton5

Para esse nível, também temos uma mensagem codificada pela **Cifra de Vigenère**, porém sem termos o conhecimento do tamanho da chave.

Para descobrir a *key* podemos usar o texto em `found1` e a ferramenta em dcode. Usando a análise estatística, obtemos uma resposta parcial: `KEYLEBGTH`.

Essa resposta é próxima de um termo legível: `KEYLENGTH`. Testando essa *key* em `found1` usando o dcode, vemos que ela decifra o texto perfeitamente.

Assim, decriptando o texto `BELOS Z` usando a *key* `KEYLENGTH` obtemos a flag.

Resposta: `RANDOM`.

Krypton6

Nesse último nível, a mensagem em `krypton7` está codificada pelo método de One-Time Pad, usando o programa `encrypt6`.

Como temos o programa, podemos usar o método do `Chosen-plaintext attack` para descobrir seu funcionamento.

Primeiro, criamos uma pasta em `/tmp/` para os arquivos desse nível

```
mkdir /tmp/files/
```

Depois, vamos criar um arquivo apenas com "A"s, para ser encriptado.

```
echo "AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA" > /tmp/files/a
```

Assim, encriptando ele por

```
./encrypt6 /tmp/files/a /tmp/files/out
```

obtemos a sequência `EICTDGYIYZKTHNSIRFXYCPFUEOCKRNEICTDGYIYZKTHNS`. Ao realizar o mesmo comando várias vezes, a resposta continua a mesma. Logo, a *key* não é aleatória.

Além disso, vemos que ela se repete produzindo a sequência `EICTDGYIYZKTHNSIRFXYCPFUEOCKRN` várias vezes.

Testando com um arquivo com apenas "B"s,

```
echo "BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB" > /tmp/files/b
./encrypt6 /tmp/files/b /tmp/files/out2
```

Obtemos a sequência `FJDUEHZJZALUIOTJSGYZDQGVFPDLSOFJDUEHZJZALUIOT`, que é exatamente a anterior com um shift a mais no alfabeto em cada caractere.

Assim, temos que o One-Time Pad não é aleatório, usa a mesma seed, cuja *key* repete ao longo da mensagem, e se comporta com shifts no alfabeto, como na cifra de Vigenère.

Por fim, um pequeno script em Python consegue recuperar a flag:

```
input = 'PNUKLYLWRQKGKBE'
key = 'EICTDGYIYZKTHNS'
```

```
base = ord('A')
flag = ''
for c_in, shift in zip(input, key):
    flag += chr( (ord(c_in) - ord(shift) - 2*base)%26 + base)

print(flag)
```

Resposta: `LFSRISNOTRANDOM`.

Outros Write-ups e arquivos

- Nada ainda

Revision #2
Created Wed, Dec 19, 2018 7:22 PM by Andrew
Updated Thu, Mar 7, 2019 8:22 PM by Andrew