Front matter

lang: ru-RU title: Lab07 author: | Anna D. Zaytseva\inst{1,3} institute: | \inst{1}RUDN University, Moscow, Russian Federation date: NEC--2024, 19 October, Moscow

Formatting

toc: false slide_level: 2 theme: metropolis header-includes:

- \metroset{progressbar=frametitle,sectionpage=progressbar,numbering=fraction}
- '\makeatletter'
- '\beamer@ignorenonframefalse'
- '\makeatother' aspectratio: 43 section-titles: true

Цель работы

Цель работы --- приобретение праккtических навыков по использованию инструмента Burp Suite.

Выполнение этапа индивидуального проекта

Step 1

Я создала функцию для генерации случайного ключа (Рис. [-@fig:001]):

```
import random
import string

def hex_key_generator(text):
    key = ''
    for i in range(len(text)):
        key += random.choice(string.ascii_letters + string.digits) # generation of a number for each character in the tex
    return key
```

{ #fig:001 width=70% }

Step 2

Поскольку операция XOR отменяет сама себя, одной функции для шифрования и для дешифрования текста будет достаточно (Рис. [-@fig:002]):

```
def encrypt_decrypt(text, key):
    new_text = ''
    for i in range(len(text)):
        new_text += chr(ord(text[i]) ^ ord(key[i % len(key)]))
    return new_text
```

{ #fig:002 width=70% }

Step 3

Я создала функцию поиска возможных ключей для текстового фрагмента (Рис. [-@fig:003]):

```
def find_possible_key(text, fragment):
    possible_keys = []
    for i in range(len(text) - len(fragment) + 1):
        possible_key = ""
        for j in range(len(fragment)):
            possible_key += chr(ord(text[i + j]) ^ ord(fragment[j]))
        possible_keys.append(possible_key)
    return possible_keys
```

{ #fig:003 width=70% }

Step 4

Проверка работы всех функций. Шифрование и дешифрование происходит корректно, как и нахождение ключей, с помощью которых можно расшифровать корректно только кусок текста (Рис. [-@fig:004]):

```
t = 'С Новым Годом, друзья!'
key = hex_key_generator(t)
encrypt = encrypt_decrypt(t, key)
decrypt = encrypt_decrypt(encrypt, key)
poss_keys = find_possible_key(encrypt, 'С Новым')
fragment = "С Новым"
print('Открытый текст: ', t, "\nКлюч: ", key, '\nШифротекст: ', encrypt, '\nИсходный текст: ', decrypt,)

print('Возможные ключи: ', poss_keys)
print('Расшифрованный фрагмент: ', encrypt_decrypt(encrypt, poss_keys[0]))
```

✓ 0.0s

```
Открытый текст:  С Новым Годом, друзья!
Ключ:  Acx3XLmtn22xq29d4aEVVk
Шифротекст:  ѠСќЙЖЇёТѣЌIцэ□□èVTΘКЙJ
Исходный текст:  С Новым Годом, друзья!
Возможные ключи:  ['Acx3XLm', 'Ъx\x10T5\x1aѪ', 'DЭw9cПA', ',ъ\x1aoѪ60', 'КЧLЖOG:', '&ѱщC>Mz', 'pt`24\rq', 'vй\x118t\x06T'
Расшифрованный фрагмент:  С Новым□OVeOЁsXгЌБЬiV□
```

{ #fig:004 width=70% }

Вывод

Приобрела практический навык по применению метода однократного гаммирования.

Библиография

- https://xakep.ru/2019/07/18/crypto-xor/
- https://bugtraq.ru/library/books/crypto/chapter7/

{.standout}

Спасибо за внимание!