CTF 101 - Cryptanalysis

SecTalks SYD0x0f

19 April 2016

#!/whoami

Pedram Hayati

- PhD (ComSci), Bsc (IT eng.)
- Partner at elttam
- Launched SecTalks non-profit meetups

Cryptography

For CTF

Why this workshop?

- There are many workshops and courses on cryptography.
- In Cryptanalysis world, the scenario is:
  - Here is the ciphertext
  - Here is the crypto algorithm
  - Here is the plaintext
  - Find a way to break it
- In CTF, this is not the case:
  - FVBJHUIYLHRJVKLHUKFVBHYLWYVBKVMPA
  - Title: I rule, Category: Crypto, Points: 50
  - What is the flag?!
- So our focus in this workshop is to teach you how to approach CTF crypto challenges.

It is not a day workshop

- Cryptography is a massive topic
- Need a lot of time to understand
- Good news: there are a lot of (free) resources from both industry and academic on learning cryptography
- We are going to focus on a tiny bit of cryptography and learn about:
    1. Crypto fundamentals
    2. Common Crypto Ciphers
    3. Cryptanalyses
    4. Breaking the crypto code using Python

Crypto fundamentals

For CTF

Have you done your homework?

- Resources from last workshop
  - 9 minutes video on cryptography 101 (by Prof. D. Brumley)
  - Introduction to Cryptography (by picoctf.com)
  - Ciphers and tools to test (from cryptool-online.org)
  - Cryptography step-by-step exercises (from Matasano)

Common ciphers

- Caesar
  - Monoalphabetic cipher
  - Key is a *k* between 1 and 25 (inclusive).
  - Replace each symbol of the plaintext with a symbol of ciphertext using a single new alphabet.

Cryptanalysis

Breaking the code

Cryptanalysis

- Analyse a cryptographic system.
- Understand cryptographic system.
- Break the system with or without knowing the key.
- Common methods

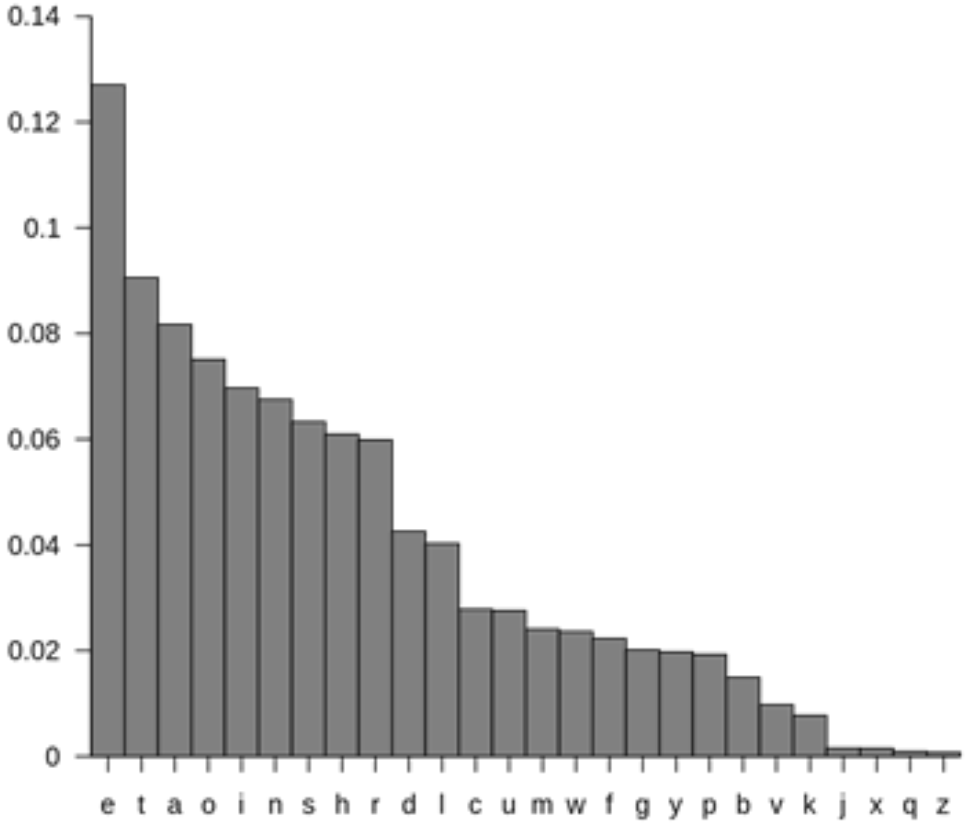
Brute-force search

- Try every possible combination of characters or data to find the key
- Works for some CTFs when the keyspace is small
- 100% guarantee of success at expense of time

Dictionary attack

- Use a dictionary of common and likely combinations
- More efficient that BF but no guarantee on finding the key

Frequency analysis



- In a plaintext, each character occurs with a characteristic frequency.
- The characteristics frequency of the plaintext may appear in the ciphertext.
- Analyse letters or group of letters
- English most common (monograms)
  - E, T, A, O, N (Etaion)
- English least common (monograms)
  - Z, Q, X

Breaking crypto

Using Python

Brute-force: Caesar cipher

FVBJHUIYLHRJVKLHUKFVBHYLWYVBKVMPA

Filename: brute-force-caesar-cipher.txt

1. Try every possible decryption key
2. Print output (sample output below)

```
Key #0: GUVF VF ZL FRPERG ZRFFNTR.
Key #1: FTUE UE YK EQODQF YQEEMSQ.
Key #2: ESTD TD XJ DPNCPE XPDDLRP.
```

How I Python?

See pythonCrashCourse.py for quick introduction to Python

Brute-force: Caesar cipher

```
5 cipher = 'FVBJHUIYLHRJVKLHUKFVBHYLWYVBKVMPA'
6 LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
```

Brute-force: Caesar cipher

```
4
5 cipher = 'FVBJHUIYLHRJVKLHUKFVBHYLWYVBKVMPA'
6 LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7
8 for key in range(len(LETTERS)):
9     plaintext = ''
```

Brute-force: Caesar cipher

```
4
5  cipher = 'FVBJHUIYLHRJVKLHUKFVBHYLWYVBKVMPA'
6  LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7
8  for key in range(len(LETTERS)):
9      plaintext = ''
10
11     for symbol in cipher:  # Loop through each character of cipher
12         if symbol in LETTERS:
13             position = LETTERS.index(symbol)  # Find symbol in alphabet
14             position = position - key  # Replaced with key previous letter
```

Brute-force: Caesar cipher

```
4
5  cipher = 'FVBJHUIYLHRJVKLHUKFVBHYLWYVBKVMPA'
6  LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'
7
8  for key in range(len(LETTERS)):
9      plaintext = ''
10
11     for symbol in cipher:  # Loop through each character of cipher
12         if symbol in LETTERS:
13             position = LETTERS.index(symbol)  # Find symbol in alphabet
14             position = position - key  # Replaced with key previous letter
15
16
17
18
19
20             # Decryption: added the letter than the key position to plaintext
21             plaintext = plaintext + LETTERS[position]
22
23         else:
24             # Symbol not find, just add cipher character with no decription
25             plaintext = plaintext + symbol
26
27     print('Key #{0}: {1}'.format(key, plaintext))
```

Brute-force: Caesar cipher

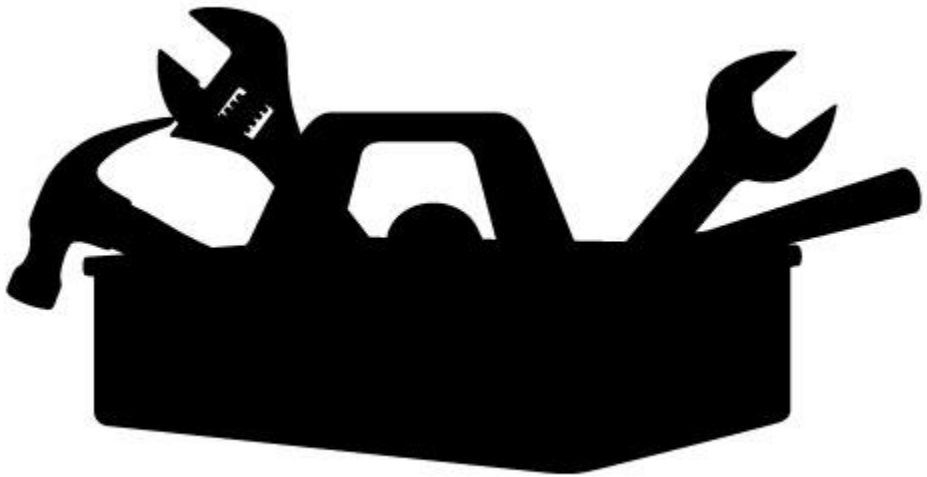
```python
"""SecTalks CTF101 - 02 - Cryptoanalysis
Brute-force Caesar cipher
"""

cipher = 'FVBJHUIYLHRJVKLHUKFVBHYLWYVBKVMPA'
LETTERS = 'ABCDEFGHIJKLMNOPQRSTUVWXYZ'

for key in range(len(LETTERS)):
    plaintext = ''

    for symbol in cipher:  # Loop through each character of cipher
        if symbol in LETTERS:
            position = LETTERS.index(symbol)  # Find symbol in alphabet
            position = position - key  # Replaced with key previous letter

            # If position is negative, add 26, length of alphabet
            if position < 0:
                position = position + len(LETTERS)

            # Decryption: added the letter than the key position to plaintext
            plaintext = plaintext + LETTERS[position]

        else:
            # Symbol not find, just add cipher character with no decription
            plaintext = plaintext + symbol

    print('Key #{0}: {1}'.format(key, plaintext))
```

Create your CTF Toolbox

Start creating it now

CTF Toolbox

- A collection of scripts, libraries and other utilities
- Quick prototyping or solving different parts of a challenge
- Add more tools over the time as you do more CTFs
- Most challenges are either being repeated or very similar.



Frequency analysis: Caesar cipher

FRPGNYXF ZRRGHCF NER NOBHG CNEGVPVCNGVAT VA VG FRPHEVGL QVFPHFFVBAF, YRNEAVAT SEBZ BGUREF, NAQ VZCEBIVAT CEBOYRZ-FBYIVAT FXVYYF. FRPGNYXF BSSREF NA NIRAHR JURER LBH PNA GRNZ-HC JVGU YVXRZVAQRQ CRBCYR GB CNEGVPVCNGR VA FBYIVAT GRPUAVPNY VG FRPHEVGL PUNYYRATRF

Filename: frequency-analysis-caesar-cipher.txt

Frequency analysis: Caesar cipher

Write a Python method to count the frequency of each cipher letter

1. Given a string, breaks it to its characters
2. Set letter = F, count = 0
3. Loop over all characters and if letter = F then count + 1
4. Print count
5. Remove letter from the string
6. If not end of string, set letter = next character in the string, count =0 and go to 3.

Frequency analysis: Caesar cipher

```
8  def get_letter_count(message):
9      """Returns a dictionary with keys of single letters and values of the
10     count of how many times they appear in the message parameter."""
11     letter_count = {}
12     for letter in message.upper():  # Uniform letters to uppercase
13         if letter in [' ', ',', '.']:
14             continue
15         if letter not in letter_count:
16             letter_count[letter] = 1
17         else:
18             letter_count[letter] += 1
19
20     return letter_count
21
```

Frequency analysis: Caesar cipher

Code a method to return most frequent letter

```
22
23 def get_most_frequent_letter(message):
24     """Return the most frequent letter in the given message dictionary"""
25     most_freq_value = 0
26     most_freq_index = None
27     for letter in message:
28         if message[letter] >= most_freq_value:
29             most_freq_value = message[letter]
30             most_freq_index = letter
31     return most_freq_index, most_freq_value
```

Frequency analysis: Caesar cipher

- The key is the number of shift between letter ‘E’ and the most frequent letter in the cipher.
- Write a code to generate this key.
- Be watchful of all possible scenarios.

Frequency analysis: Caesar cipher

```
41 e_position = LETTERS.index('E')
42 m_position = LETTERS.index(most_frequent_letter[0])
43 key = m_position - e_position
44
45 if key < 0:
46     key = key + len(LETTERS)
47 print(key)
48
```

Frequency analysis: Caesar cipher

- Now, same as the second loop in bruteforce code, loop through all symbols and decrypt the cipher.

```
49 for symbol in cipher:
50     try:
51         position = LETTERS.index(symbol)
52         position = position - key
53         if position < 0:
54             position = position + len(LETTERS)
55         plaintext = plaintext + LETTERS[position]
56     except:
57         plaintext = plaintext + symbol
58
59 print(plaintext)
```

Ngrams

- Human language carries also frequent 2 letters, 3 letters word. For example in English:
  - Top bigrams: TH, HE, IN, ER, AN
  - Top trigrams: THE, AND, ING, ENT
- Similar to frequency analysis of monograms, counting the frequency of bigrams (2 consecutive characters), trigrams (3 consecutive characters, etc. helps to understands the ciphertext and find clues to break it.
- More information
  - http://practicalcryptography.com/cryptanalysis/letter-frequencies-various-languages/english-letter-frequencies/

Frequency analysis: ngrams

- Write a new Python method to:

1. Get an input string and the size of ngram (e.g. 2, 3, etc.)
2. Count ngrams of the input string for all the ngrams smaller or equal to the given ngram size
3. Sort and print the output from top most frequent ngram to least.

Note: Write your own method and do not use NLTK/NLP libs.

Homework

1. Write a code that generates Caesar cipher for all possible keys. Test the output of your program in brute-force program.
2. Write a better frequency analysis scripting by using “frequency match score” algorithm.

Next

What will be covered in the next workshop

Coming up

Modern cryptanalysis

- Modern cryptography
  - Single-byte XOR
  - Multi-byte XOR
- Breaking the crypto code using Python
- Some crypto challenges

That’s all for now

Feel free to get in touch if you have any question:

pi3ch@irc.sectalks.org #sectalks
pedram@elttam.com.au

References

1. Modern Cryptanalysis: Techniques for Advanced Code Breaking
2. Understanding Cryptology: Cryptanalysis, Dr. Kerry McKay, OpenSecurityTraining.info
3. https://www.khanacademy.org/computing/computer-science/cryptography
4. https://inventwithpython.com/hacking/chapter7.html
5. https://picoctf.com/crypto_mats/
6. http://practicalcryptography.com/cryptanalysis/

Copyright