Python with Applications, PIC16
E. Ryu
Spring 2019

Midterm Exam
Friday, May 3, 2019
50 minutes, 2 questions, 100 points, 3 pages

While we don't expect you will need more space than provided,
you may continue on the back of the pages.
This exam is closed book and closed notes.

Do not turn to the next page
until the start of the exam.

1. (50 points) When a scholar publishes a research paper, a unique Document Object Identifier (DOI), whic looks something like is assigned to the publication. `10.1109/TIT.2005.862083` is an example. See `https://en.wikipedia.org/wiki/Digital_object_identifier`

   Write a program that extracts the DOI of the publication from its webpage. For simplicity, we have downloaded the webpages into HTML files. Instead of accessing the webpages over the internet, do

   ```
   with open('boyer.html','r') as reader:
       webpage = reader.read()
   ```

   Download the starter code `midterm_p1_starter.py` and the test files `midterm_p1_files.zip`. Extract the 11 DOIs from the 11 publication web pages in `midterm_p1_files.zip`.

   Hint. HTML files are basically webpages written in code; it contains the webpage's content and information on how it should be formatted. By double clicking the HTML file, you can open it in a browser and see what it should approximately look like.

2. (50 points) You are a hacker, and you wish to carry out a *ransomware* attack to extort (steal) money from innocent people. This attack is carried out in three steps:

   1. (attacker→victim) The attacker generates a public-private key pair and places the public key in a malicious program (virus). The attacker manipulates the victim to have him/her run this malicious program.
   2. (victim→attacker) The malicious program generates a random symmetric key and encrypts the victim's data with it. It uses the public key to encrypt the symmetric key. The malicious program deletes the symmetric key and the original (unencrypted) data to prevent recovery. The victim is left with encrypted data and an encrypted symmetric key. The program displays the message that if the victim sends the encrypted key and ransom (money) to the attacker, then the attacker will send back the decrypted symmetric key which the victim can use to decrypt the data.
   3. (attacker→victim) The attacker receives the payment, decrypts the key with the attacker's private key, and sends the symmetric key to the victim. The victim decrypts the data with the needed symmetric key thereby completing the ransomware attack.

   Download the starter code `midterm_p2_starter.py` and the test files `midterm_p2_files.zip`. Implement this attack in the following three functions:

```
encrypted_key = ransomAttack(public_key,filenames)
```

   generates a random key using `Crypto.Random.get_random_bytes(64)` and encrypts (using symmetric-key encryption) and overwrites all files with names in the list `filenames`. It `return`s the encrypted symmetric key using `public_key` (with public-key encryption). After running this function, the files should no longer contain the original content.

```
decrypted_key = recoverKey(key_pair,encrypted_key)
```

   uses the `key_pair` (available to the attacker) to decrypt the symmetric key.

```
ransomRelease(decrypted_key,filenames)
```

   uses the `decrypted_key` (which the victim received from the attacker after paying money) to decrypt and overwrite all files with names in the list `filenames`. After running this function, the original content of the files should be restored.

   Remark. Generate the public-private key pair and the symmetric key with

```
RSA.generate(1024)
```

```
Crypto.Random.get_random_bytes(64)
```

   which means the length of this symmetric key is `64`. (If you use a symmetric key length longer than `64`, you need to also use a key length longer than `1024` for the RSA object.)

   Remark. In `ransomAttack`, overwrite files with encrypted content in hexadecimal format with

```
with open('filename.txt','w') as writer:
    writer.write(encrypt_obj.encrypt(text).encode('hex'))
```

   In `ransomRelease`, read the encrypted content in hexadecimal format with

```
with open('filename.txt','r') as reader:
    text = reader.read().decode('hex')
```

Python Regex Cheatsheet

Regular Expression Basics

.	Any character except newline
a	The character a
ab	The string ab
a|b	a or b
a*	0 or more a's
\	Escapes a special character

Regular Expression Quantifiers

*	0 or more
+	1 or more
?	0 or 1
{2}	Exactly 2
{2, 5}	Between 2 and 5
{2,}	2 or more
(,5}	Up to 5

Default is greedy. Append ? for reluctant.

Regular Expression Groups

(...)	Capturing group
(?P<Y>...)	Capturing group named Y
(?:...)	Non-capturing group
\Y	Match the Y'th captured group
(?P=Y)	Match the named group Y
(?#...)	Comment

Regular Expression Character Classes

[ab-d]	One character of: a, b, c, d
[^ab-d]	One character except: a, b, c, d
[\b]	Backspace character
\d	One digit
\D	One non-digit
\s	One whitespace
\S	One non-whitespace
\w	One word character
\W	One non-word character

Regular Expression Assertions

^	Start of string
\A	Start of string, ignores m flag
$	End of string
\Z	End of string, ignores m flag
\b	Word boundary
\B	Non-word boundary
(?=...)	Positive lookahead
(?!...)	Negative lookahead
(?<=...)	Positive lookbehind
(?<!...)	Negative lookbehind
(?()|)	Conditional

Regular Expression Flags

i	Ignore case
m	^ and $ match start and end of line
s	. matches newline as well
x	Allow spaces and comments
L	Locale character classes
u	Unicode character classes
(?iLmsux)	Set flags within regex

Regular Expression Special Characters

\n	Newline
\r	Carriage return
\t	Tab
\YYY	Octal character YYY
\xYY	Hexadecimal character YY

Regular Expression Replacement

\g<0>	Insert entire match
\g<Y>	Insert match Y (name or number)
\Y	Insert group numbered Y