CMSC 426 – Principles of Computer Security

Spring 2019

Lab 3 – Cryptanalysis

Full name: Daniel Roh
UMBC username: DRoh1

Part 1: Cryptanalysis

1A. Analyze the hexdump of the **`encrypted.enc`** file. What mode of operation did the block cipher most likely use? What block size was used to encrypt? Justify your answers.
[7 points]

Mode: ECB
Block Size: 64

- Due to the amount of repeated data in the hexdump, the repeats give a idea that ECB is used as each text block is being encrypted with the same key.
- In the hex dump, it seems that a lot of the data is repeated in groups of 8. Since each hex group is 8 bits, then there can be 8x8=64 bits.

1B. Based on the block size, which block cipher (DES, 3DES, or AES) was used to encrypt the **`encrypted.enc`** file? Justify your answer.
[3 points]

- DES was most likely used to encrypt this file since the hexdump shows a block size of 64 bytes and DES is the one type of encryption that has a block size of 64 bits. While 3DES also has a block size of 64 bytes, I ruled out 3DES since we are only looking for one key.

1C. What file format do you think the original file was in? Justify your answer. If you think there is more than one possibility, make sure to justify each of them.
[3 points]

- File type = .doc

- The .doc file produced a hexdump that has similar hex spacings to the encrypted file

1D. Describe how the seed is being computed inside the **`keygen.py`** file. Is this a cryptographically secure method? Why or why not?
[3 points]

- The seed is being calculated by the current time ANDed to the hex value FFFFFF
- This is not a cryptographically secure method as the “current time” is normally recorded when a message is sent to another user as a timestamp and can be spoofed to make it look like the time it was actually sent to get the key back.

1E. What crucial piece of information regarding the encrypted files will allow you to attack the key generation function? Outline an attack on the key generation function.
[7 points]

- Since we know the rough time and day that Dr. Gibson sent the message. We can then run a brute force method to gain all the keys that were possibly used to encrypt the message.
- Starting at the first possible time that Dr. Gibson sent the message, we can generate a key. Then move time forward and generate the key again. We can keep doing this until we have a table of all possible keys that was used to encrypt the file.

1F. Your brute force attack will produce thousands of possible decryptions. Describe the method you will use to narrow down the possibilities.
[3 points]

- Since we know the general file type is a doc, we can use a standard file to look for the output when testing the decryption. In this case, we can generate a key and check if the decrypted file matches of parts of the standard file. If a match is found, the we can save it to a file for later inspection.

1G. What is the decrypted message? [3 points]

From:

http://pyrrhiccomedy.tumblr.com/post/113417212152

I still have super strong feelings about dogs as a concept. I mean we take these animals and we shape them into the perfect companions for our species in a process that takes tens of thousands of years. We make them so that they love us, and serve us, and will learn how to do whatever we need them to do. We make them want to guard us while we sleep, to guard our livestock, to guard our homes, to help us hunt, to act as beasts of burden, to watch our children and our fields. They'll learn how to see for us. They'll run into burning buildings for us.

Dogs walked beside us every step of the way as we became the dominant species on the planet. And all they ever ask for in return is enough food to stay alive and as much love as we will give them. And I am heartened and humbled to know that, when humanity set about the daunting and monumental task of creating another race to be our companions on Earth, we made something as noble and selfless and loving as dogs.

In an existential and anthropological sense, they're humanity's children. We gotta be nice to dogs. We owe them that much.

1H. Turn in your `lab3_1.py` file separately and make sure to fill out the header at the top. [25 points]

[Submitted as separate file (do not paste the contents here)]

Part 2: Implementing RSA

2A. *In your own words* (without being exhaustive), describe how Fermat's primality test works to effectively test primality. You'll probably want to read the Wikipedia page.
[3 points]

- Fermat's primality test checks by picking random numbers for a number of times and testing to see if the random number has a common denominator or if the random value powered to the tested number minus 1 has a remainder other than 1.

2B. What value of **`max_iter`** did you choose to use for Fermat's primality test? Why?
[3 points]

max_iter = 100

- I choose this value because since most common divisors are divisible by 100, If a value happens to be bigger then 100, there is a high chance that the value can be checked for primality in about 100 tries.

2C. Turn in your **`lab3_2.py`** file separately and make sure to fill out the header at the top.
[25 points]

[Submitted as separate file (do not paste the contents here)]

2D. Turn in your **`typescript`** file separately. Make sure it is complete.
[10 points]

[Submitted as separate file (do not paste the contents here)]