14741/18631 HW1

Hao Zhang

TOTAL POINTS

87 / 100

QUESTION 1

1 Breaking Vigenere **14 / 15**

Understanding

✓ **+ 5 pts Derivation of key length - Kasiski Examination / Index of Coincidence**

+ **5 pts** Derivation of key length - brute force

✓ **+ 5 pts Derivation of actual key - frequency analysis**

+ **10 pts** Performed other forms of reasonable analysis on the ciphertext and obtained the key

✓ **- 1 pts Mentioned the attacks but without sufficient explanation**

+ **0 pts** Did not mention any attack techniques

CTF Problem

✓ **+ 1 pts CTF Username**

✓ **+ 2 pts Flag**

Others

+ **2 pts** Implemented own solver

✓ **+ 2 pts Used online solver with proper citation**

+ **1 pts** Mentioned online solver without citation

- **1 pts** Formatting issue

How does the frequency solver derive the key?

QUESTION 2

2 One-time pads **20 / 20**

Vulnerability

✓ **+ 8 pts Good understandings**

+ **6 pts** Slightly incorrect explanations

+ **4 pts** Insufficient explanations

+ **0 pts** No explanation

Exploit

✓ **+ 8 pts Good understandings**

+ **6 pts** Slightly incorrect explanations

+ **4 pts** Insufficient explanations

+ **0 pts** No explanation

CTF Problem

✓ **+ 4 pts Flag**

- **2 pts** Missing username

+ **0 pts** Incorrect/Missing flag

Others

- **1 pts** Extra Page

- **2 pts** Format not followed

- **1 pts** Flag should be at the top

QUESTION 3

3 ECB **24 / 25**

Vulnerability

✓ **+ 10 pts Good understandings**

+ **8 pts** Slightly incorrect explanations

+ **4 pts** Insufficient explanations

+ **0 pts** No explanation

✓ **- 1 pts Didn't mention the blocks are encrypted independently**

Exploit

✓ **+ 10 pts Good understandings**

+ **8 pts** Slightly incorrect explanations

+ **4 pts** Insufficient explanations

+ **0 pts** No explanation

CTF Problem

✓ **+ 5 pts Flag**

- **1 pts** Missing username

Others

- **1 pts** Formatting issue

QUESTION 4

Padding attack 40 pts

4.1 Decryption attack 16 / 20

Vulnerability

✓ + 2 pts **Mentions padding oracle LEAKS INFORMATION as a vulnerability**

+ 2 pts Mentions program DOES NOT CHECK CIPHERTEXT INTEGRITY as a vulnerability

+ 0 pts Mentioning Padding oracle Attack is not the same

Understanding of Decryption Attack

+ 8 pts Correct understanding of Decryption Attack

+ 6 pts Partial Credit A: correct understanding of last-block decryption

✓ + 6 pts **Partial Credit C: Correct Understanding of Decryption attack but needs to elaborate more**

+ 3 pts Partial Credit B: correct understanding of last-byte decryption

+ 3 pts Partial Credit D: Understands basics of attack but has provided no explanation/Needs more explanation

+ 0 pts Not Attempted

Decryption Code

✓ + 8 pts **Good decryption code with correctly recovered plaintext**

+ 6 pts Partial Credit A: code only recovers one block

+ 2 pts Partial Credit B: code only recovers last byte

+ 0 pts Not Attempted / Incorrect

4.2 Encryption attack 13 / 20

Understanding of Re-encryption Attack

+ 10 pts Solid understanding of Re-encryption Attack

+ 6 pts Partial Credit A: understanding of re-encryption attack but lack detail in explanation

+ 3 pts Partial Credit B: understanding of only how to modify plaintext

✓ + 3 pts **Partial Credit C: Needs to elaborate**

+ 0 pts Not Attempted / No Explanation given

Re-encryption Code

✓ + 10 pts **Working Re-encryption Code + Flag**

+ 8 pts Partial Credit A: working code but no flag

+ 6 pts Partial Credit B: code attempts to modify plaintext in correct way but does not work as a whole

+ 0 pts Not Attempted

Others

- 1 pts Missing username

- 2 pts Formatting issue

- 2 pts Did not upload code to separate assignment

- 2 pts Did not paste code as was instructed

- 1 pts Pages need to be mapped accurately

- 2 pts Too many Pages

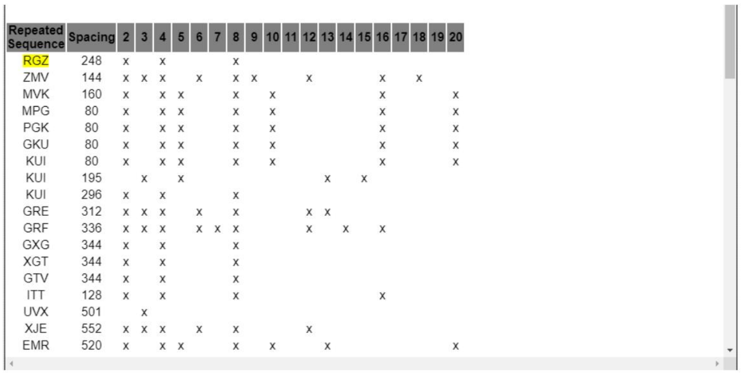
Question 1

CTF Username: IwishIcansing   Flag: 1ee7a245d129fa6075eb9d02909cdd91

1. Explain the vulnerability in the program, and explain conceptually how that vulnerability can be exploited to get the flag.
   The vulnerability of this program is that the Vigenere's key is too short, therefore, in order to encrypt the entire message, the key has to repeat itself many times, which forms a certain pattern. A solver can take a random group of adjacent three letters "XYZ", the distance[1] between X, Y, Z can be calculated. For any key length N, "ABC", which is K*N spacing apart from XYZ, has to have the same distance apart from each others. By collecting a large amount of data of grouping like XYZ and their spacing away to ABC, a solver can determine the possible key length. After the key length is found, we can extract all letters that are N distance[2] apart from each other. Since they used the same key for encryption, a frequency solver can be used to determine the actual key.
2. How did you exploit the vulnerability?

a. Since the prompts say someone is talking at 192.168.2.83:1111. A simple netcat command would establish connection with the port and listen to the message at the port. The command I used was: nc 192.168.2.83 1111.

b. Then, I got the encrypted message. I plugged it into a Vigenere solver. The solver lists many three adjacent letter groups that have an identical repetition with certain spacing apart. For example, for a grouping RGZ, the solver finds another RGZ that is 248 characters spacing apart. Therefore, a possible key length can be any factor of 248. By looking at all the possible key lengths, we find the most likely one based on how many times it is considered possible key length. This uses a similar approach as what I described in 1, which both exploits the fact that the key length is not long enough, so after the letters that are N spacing apart use the same key for encryption.



Repeated Sequence	Spacing	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
RGZ	248	x		x				x												
ZMV	144	x	x	x		x		x	x			x				x		x		
MVK	160	x		x	x			x		x						x				x
MPG	80	x		x	x			x		x						x				x
PGK	80	x		x	x			x		x						x				x
GKU	80	x		x	x			x		x						x				x
KUI	80	x		x	x			x		x						x				x
KUI	195		x		x								x		x					
KUI	296	x		x				x												
GRE	312	x	x	x		x		x				x	x							
GRF	336	x	x	x		x	x	x				x		x		x				
GXG	344	x		x				x												
XGT	344	x		x				x												
GTV	344	x		x				x												
ITT	128	x		x				x								x				
UVX	501		x																	
XJE	552	x	x	x		x		x				x								
EMR	520	x		x	x			x		x			x							x

c. From the picture above, I found that 8 is the most frequent (2 and 4 are 8's factor, can be considered later). I set 8 as the key length. Then the solver will extract all letters once every 8 letters. A simple frequency solver will show what the actual key is.[3]

[1] for example, C is 2 distance away from E, A is 7 distance away from H
[2] For ABCDAXXX, here, A is 4 distance from another A.
[3] Solver used: http://www.brianveitch.com/maze-runner/frequency-analysis-vigenere/
Tutorial of how to use the solver: youtube.com/watch?v=P4z3jAOzT9I

1 Breaking Vigenere 14 / 15

Understanding

✓ **+ 5 pts** Derivation of key length - Kasiski Examination / Index of Coincidence

+ 5 pts Derivation of key length - brute force

✓ **+ 5 pts** Derivation of actual key - frequency analysis

+ 10 pts Performed other forms of reasonable analysis on the ciphertext and obtained the key

✓ **- 1 pts** Mentioned the attacks but without sufficient explanation

+ 0 pts Did not mention any attack techniques

CTF Problem

✓ **+ 1 pts** CTF Username

✓ **+ 2 pts** Flag

Others

+ 2 pts Implemented own solver

✓ **+ 2 pts** Used online solver with proper citation

+ 1 pts Mentioned online solver without citation

- 1 pts Formatting issue

How does the frequency solver derive the key?

Question 2

CTF Username: IwishIcansing

Flag:65084690733926391919095995899419

1. Explain the vulnerability in the program, and explain conceptually how that vulnerability can be exploited to get the flag.

The vulnerability here is that the key for one time pad is reused. Since the key is reused. We can exploit it as below. Let M1 be the original plaintext with the flag, and the program used its unchanging key K to encrypt it to C1. We can then randomly enter a plaintext M2 for encryption, which returns C2. With M2, C1, C2. By math, we can obtain M1 with the equations below.

M1 XOR K = C1
M2 XOR K = C2
C1 XOR C2 = M1 XOR M2 -.> M1 = (C1 XOR C2) XOR M2

2. How did you exploit the vulnerability?

Note: M1, M2, C1, C2 below refers to the variable in the equation above.

a. Using netcat to listen to the message just like question 1
b. After entering into its interface, enter 1 into the encryption interface.
c. Randomly put in a 32 length message (M2). The server will generate the ciphertext C2
d. Convert M2 of 32 character length into hex with this online hex to text converter:https://www.online-toolz.com/tools/text-hex-convertor.php
e. Now, enter 2 in the interface to get stored ciphertext C1
f. Use online XOR calculator:http://xor.pw/# to compute (C1 XOR C2) XOR M2
g. The value we return is M1, convert it to ASCII to get actual flag

2 One-time pads **20 / 20**

Vulnerability

- ✓ **+ 8 pts Good understandings**
- **+ 6 pts** Slightly incorrect explanations
- **+ 4 pts** Insufficient explanations
- **+ 0 pts** No explanation

Exploit

- ✓ **+ 8 pts Good understandings**
- **+ 6 pts** Slightly incorrect explanations
- **+ 4 pts** Insufficient explanations
- **+ 0 pts** No explanation

CTF Problem

- ✓ **+ 4 pts Flag**
- **- 2 pts** Missing username
- **+ 0 pts** Incorrect/Missing flag

Others

- **- 1 pts** Extra Page
- **- 2 pts** Format not followed
- **- 1 pts** Flag should be at the top

Question 3

CTF Username: IwishIcansing
Flag - 638f0d137d93b3b484da89b73f5acd21

1. Explain the vulnerability in the program, and explain conceptually how that vulnerability can be exploited to get the flag.

ECB has the vulnerability that the same plaintext with the same key will end up with the same ciphertext. This can be exploited by comparing ciphertexts. If we see the same chunk of ciphertext, we can infer that a chunk of plaintext is the same.

2. How did you exploit the vulnerability?

a. I first started reading the code. I find out the code will encrypt an unknown plaintext(M1) and a known plaintext (M2) into ciphertext (C1,C2) in hex and display it to me. Then, it asks me to input ciphertext (C3). The system will decrypt it into M3. If the M3 match a string starting with I am yes an admin, the flag can be obtained.
b. Then I ran this program in putty, which returned a C1 and a C2. I used an online tool to compare two ciphertexts. I found out C1 and C2 are exactly the same at the middle portion, but differ at the beginning and end.
c. Based on the prompt "Here is an admin cookie:" for C1. I guessed it might be that the second ciphertext (C2) has the wrong beginning as I am not an admin.
d. Therefore, I tried copying C1 and sending it as my cookie C3. The reason in doing it is that ECB uses the same key for encryption and decryption. So I can obtain M1 from C1. I see M1 is "I am yes an administrator. This cookie expires 2010-01-01.......". Here, I know that the beginning part of my cipher is current, but I need to adjust the expiration date.
e. So I copied the last part of the C2, which has the expiration date at 2022, and used it to replace C1's end. Now, I am able to capture the flag. So this is what is bad about ECB. By seeing the plaintext and ciphertext, I am able to easily identify the association of which block of plaintext corresponds to which block of ciphertext, which is like getting an one-to-one mapping between plaintext and ciphertext. Therefore, when I am given a ciphertext, I am able to chunk the ciphertext into different blocks, and find the plaintext for each block. Key takeaway: ECB is bad and so easy to hack. While I was studying ECB, I find a good website that shows how easy it is to exploit ECB(https://notsosecure.com/hacking-crypto-fun-profit/). It really opens up my mind.

3 ECB **24 / 25**

Vulnerability

✓ **+ 10 pts** Good understandings

+ 8 pts Slightly incorrect explanations

+ 4 pts Insufficient explanations

+ 0 pts No explanation

✓ **- 1 pts** Didn't mention the blocks are encrypted independently

Exploit

✓ **+ 10 pts** Good understandings

+ 8 pts Slightly incorrect explanations

+ 4 pts Insufficient explanations

+ 0 pts No explanation

CTF Problem

✓ **+ 5 pts** Flag

- 1 pts Missing username

Others

- 1 pts Formatting issue

Question 4

CTF Username: IwishIcansing
Flag - 457fa2dd43bb36e1be04d9dfc519ee59

See my teammate’s submission:
His name: Zhichao Chen
His andrew ID: zhichaoc

4.1 Decryption attack **16 / 20**

Vulnerability

✓ **+ 2 pts** **Mentions padding oracle LEAKS INFORMATION as a vulnerability**

+ 2 pts Mentions program DOES NOT CHECK CIPHERTEXT INTEGRITY as a vulnerability

+ 0 pts Mentioning Padding oracle Attack is not the same

Understanding of Decryption Attack

+ 8 pts Correct understanding of Decryption Attack

+ 6 pts Partial Credit A: correct understanding of last-block decryption

✓ **+ 6 pts** **Partial Credit C: Correct Understanding of Decryption attack but needs to elaborate more**

+ 3 pts Partial Credit B: correct understanding of last-byte decryption

+ 3 pts Partial Credit D: Understands basics of attack but has provided no explanation/Needs more explanation

+ 0 pts Not Attempted

Decryption Code

✓ **+ 8 pts** **Good decryption code with correctly recovered plaintext**

+ 6 pts Partial Credit A: code only recovers one block

+ 2 pts Partial Credit B: code only recovers last byte

+ 0 pts Not Attempted / Incorrect

Question 4

CTF Username: IwishIcansing
Flag - 457fa2dd43bb36e1be04d9dfc519ee59

See my teammate's submission:
His name: Zhichao Chen
His andrew ID: zhichaoc

4.2 Encryption attack **13 / 20**

Understanding of Re-encryption Attack

- **+ 10 pts** Solid understanding of Re-encryption Attack
- **+ 6 pts** Partial Credit A: understanding of re-encryption attack but lack detail in explanation
- **+ 3 pts** Partial Credit B: understanding of only how to modify plaintext
- ✓ **+ 3 pts** **Partial Credit C: Needs to elaborate**
- **+ 0 pts** Not Attempted / No Explanation given

Re-encryption Code

- ✓ **+ 10 pts** **Working Re-encryption Code + Flag**
- **+ 8 pts** Partial Credit A: working code but no flag
- **+ 6 pts** Partial Credit B: code attempts to modify plaintext in correct way but does not work as a whole
- **+ 0 pts** Not Attempted

Others

- **- 1 pts** Missing username
- **- 2 pts** Formatting issue
- **- 2 pts** Did not upload code to separate assignment
- **- 2 pts** Did not paste code as was instructed
- **- 1 pts** Pages need to be mapped accurately
- **- 2 pts** Too many Pages