

**Electrical and Computer Engineering Department**

**ENCS4320**

**APPLIED CRYPTOGRAPHY**

**HW # 1**

**Padding Oracle Attack Lab**

**Instructor: Dr. Mohammad Hussein.**

**Section: 2.**

**Student Name: Saif Battah | 1170986**

**Student Name: Khaled Roub | 1192941**

**Date: Sunday 02/07/2023.**

Abstract

In this lab we will learn how to attack weaknesses and obtain hidden messages using a Cryptography Padding Oracle Attack and to gain a practical experience. Two messages are encrypted and hidden behind distinct port numbers within a container. The lab contains 3 tasks, Task 1 is a simple AES-128-CBC encryption decryption process on 5, 10, 16 byte messages.

Task 2 relates to get a secret message from port 5000, and Task 3 refers to get a secret message from port 6000. The lab leads you through the process of launching the attack, decrypting the cipher-text, and extracting the plaintext from the secret messages. Participants develop practical skills in finding vulnerabilities and conducting padding oracle attacks in real-world scenarios by completing this lab.

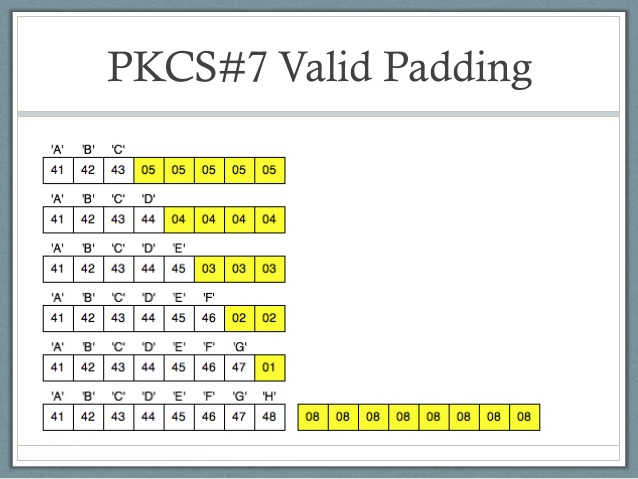
Introduction

**Padding in General**

The padding oracle attack is a strong cryptographic attack that takes advantage of faults in the validation of padding in encrypted data. An adversary can use this approach to decode encrypted data without having the encryption key or any knowledge of the particular encryption technique utilized. The secrecy of encrypted data can be damaged by expertly modifying cipher texts and watching the replies from a padding oracle, potentially resulting in sensitive information leaking.

Block ciphers, such as the frequently used CBC (Cipher Block Chaining) mode, sometimes need plaintext messages to be padded or extended to fit with the cipher's block size. In reality, the most popular padding methods are PKCS#5 and its successor PKCS#7. To ensure compatibility with the block cipher, these padding techniques add a specified pattern of bytes to the plaintext. Before returning the plaintext message, the decryption procedure often entails checking and eliminating this padding.

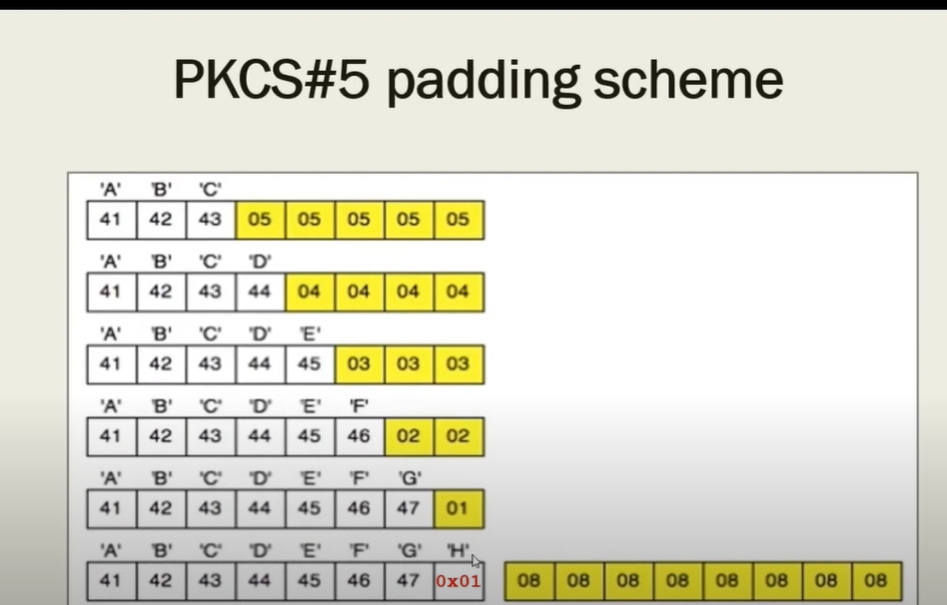
Even so, the vulnerability occurs when a server or application working as a padding oracle responds to padding validation problems in a different way. Instead of a general "decryption failed" error, it reveals information regarding the padding's validity. An attacker can use this behavior to decode cipher-texts and, in some situations, encrypt arbitrary data without knowing the encryption key.



*PKCS#7 Padding*

**PKCS#5**

PKCS#5, or Public Key Cryptography Standards #5, is a common padding technique in cryptographic systems. It is mostly used in block ciphers to guarantee that plaintext messages are appropriately aligned with the block size. To achieve the appropriate block size, PKCS#5 padding includes appending a predefined pattern of bytes to the end of a plaintext message. The number of padding bytes added is represented by the value of each additional byte. For example, if three bytes are required for padding, each byte will be 0x03. The padding is checked and deleted during decryption to get the original plaintext.



*PKCS#5 Padding*

However, if the server or application leaks knowledge about the padding validation process, PKCS#5 padding is subject to padding oracle attacks. To prevent against such attacks, it is critical to adopt suitable security measures and maintain the integrity of the padding oracle.

Let's say we have a block cipher with a block size of 8 bytes (64 bits) and we want to encrypt the plaintext message "HELLO" using PKCS#5 padding.

1. The plaintext "HELLO" is 5 bytes in length, which is less than the block size. So, padding is required to reach the block size.
2. Since we need to add 3 bytes of padding, each padding byte will have a value of 0x03.
3. The padded plaintext becomes "HELLO\x03\x03\x03".
4. The block cipher encrypts the padded plaintext using the chosen encryption algorithm and key.

Procedure

**Task 1:** Getting familiar with padding

In this task we are going to use AES-128-CBC Encryption & Decryption techniques.

We will work on 5, 10 and 16 byte messages.

As mentioned in the lab we are going to do the following steps:

* echo -n "xxxxxx" > filename.  
   This command will write "xxxxx" to file called "filename" without a newline.
* openssl enc -aes-128-cbc -e -in P -out C

This command will do an AES-128 Encryption of model CBC "-e" means encryption, "-in" the input file for encryption operation and contain plaintext, "-out" the output file of encryption operation contains the cipher-text.

* openssl enc -aes-128-cbc -d -nopad -in C -out P\_new

The same as above but "-d" means decryption, "-nopad" to remove padding.

* xxd PT1\_new

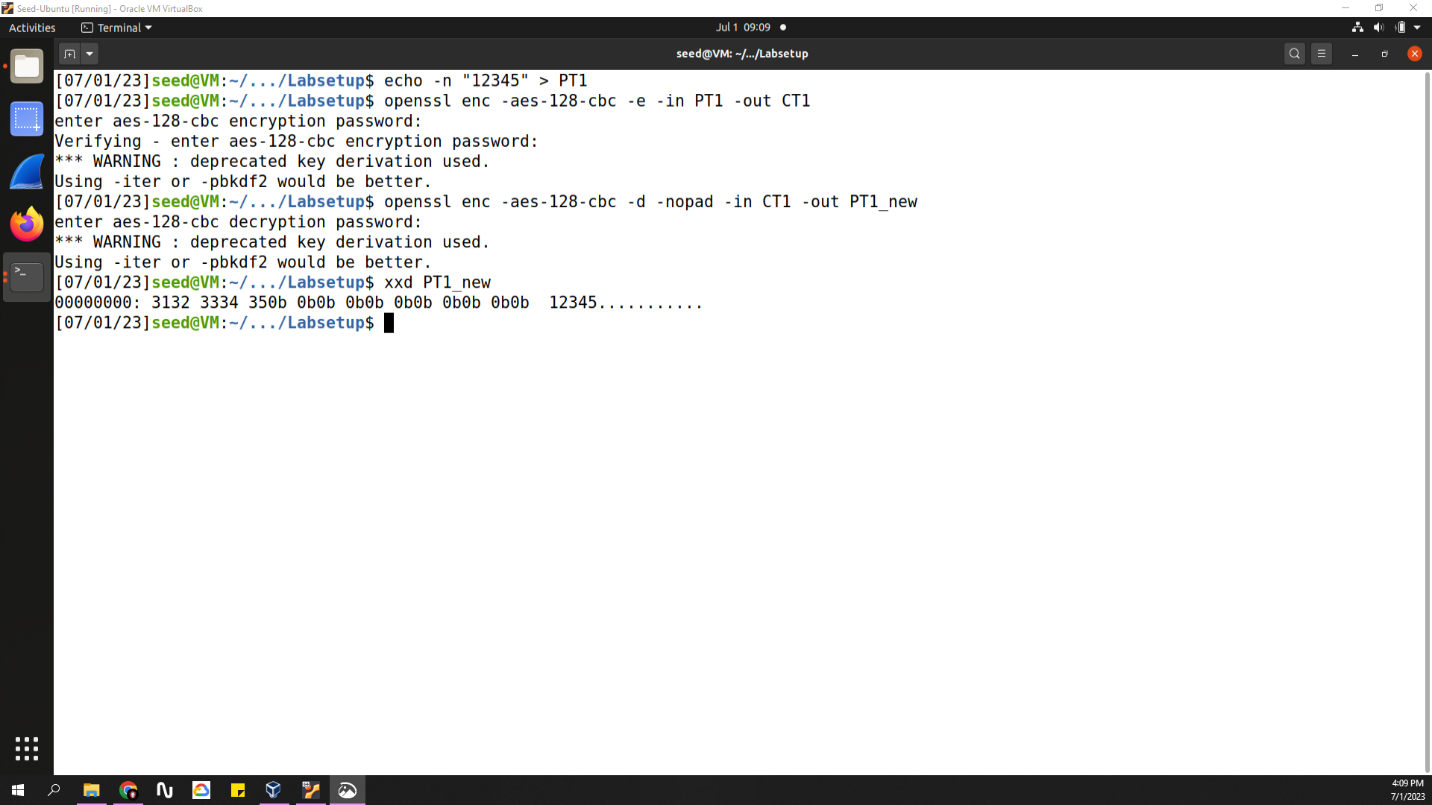
This to print the decrypted cipher text in hexadecimal format.

For 5 byte -> 12345

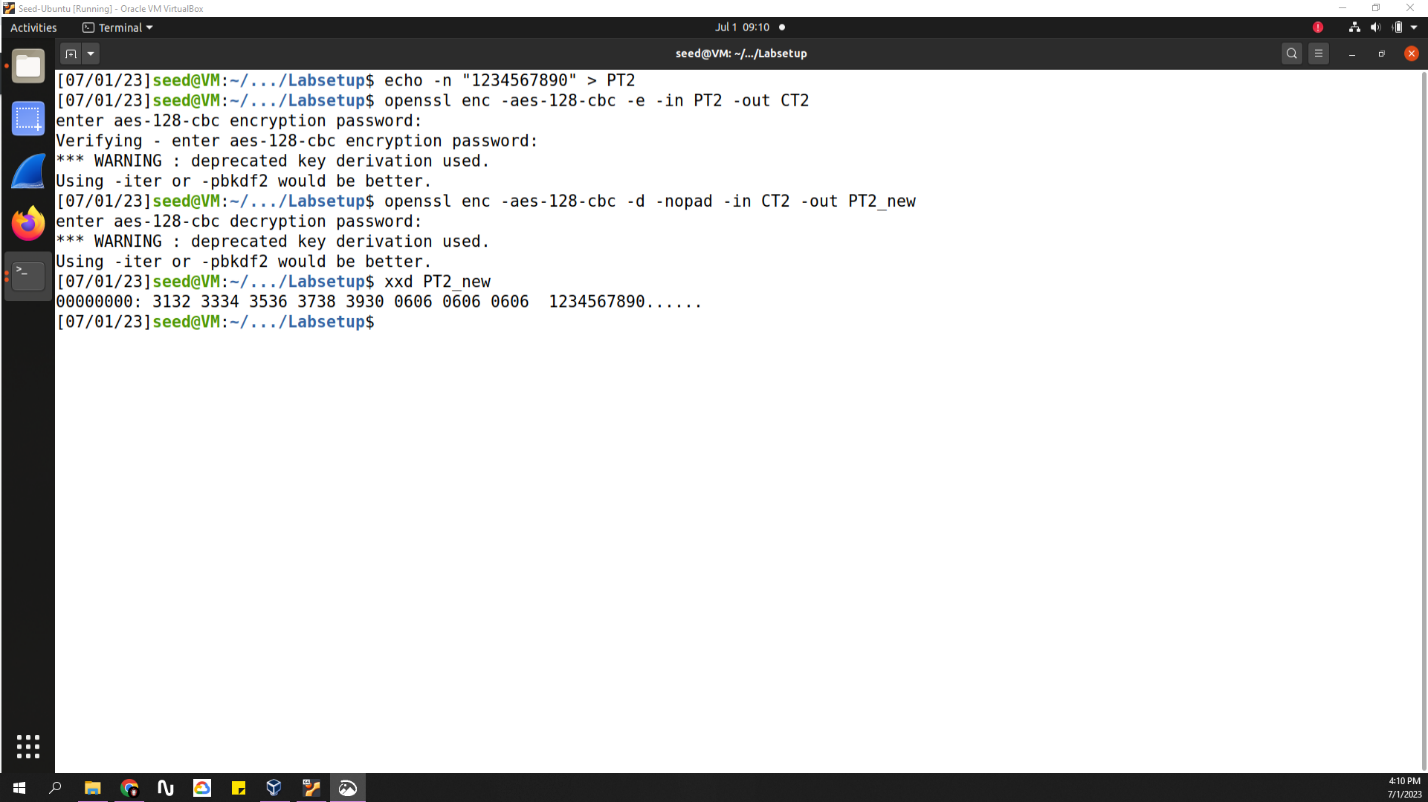
For 10 byte -> 1234567890

For 16 byte -> 1234567890AAAAAA

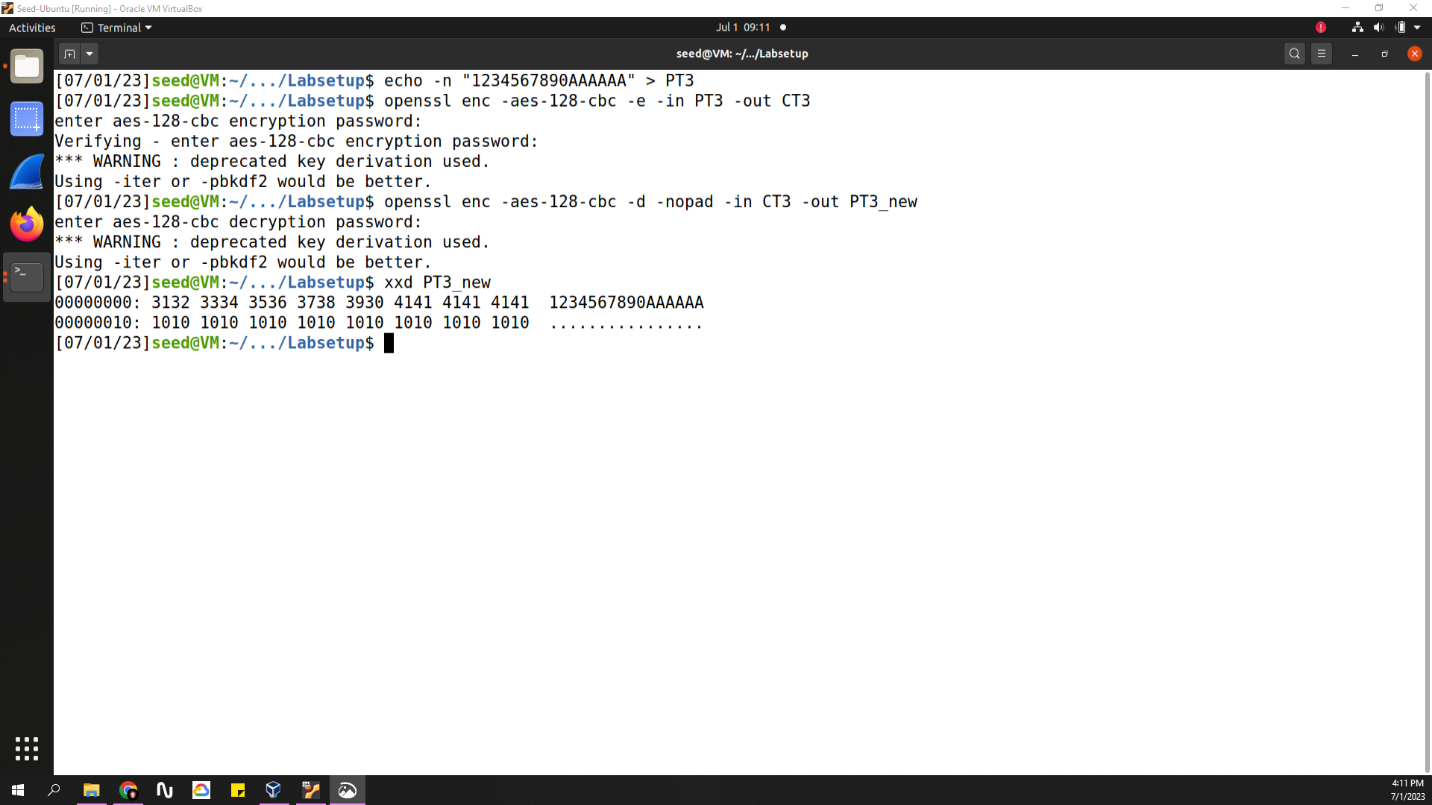
1. 5 byte message:



1. 10 byte message:



1. 16 byte message:



As shown in the figures above we've append "12345" to PT1, "1234567890" to PT2 and "1234567890AAAAAA" to PT3.

Then we encrypt each one using -e and save the result in CT [1, 2, 3].

After that we decrypt the CT [1, 2, 3] using –d and save the decrypted message in PT [1, 2, 3] \_new.

Finally we printout the PT [1, 2, 3] \_new contents in hexadecimal format.

As shown the result of PT [1, 2, 3] \_new is the same of PT [1, 2, 3].

So our Enc/Dec is Done Successfully!

**Task 2:** Padding oracle attack Lvl1

In this task we will do an oracle attack on 10.9.0.80 on port 5000, this address have a hidden '48' byte message within it, so we grab it to make the attack on it.

This message is cipher it also contains the IV within it.

IV 🡪 first 16 byte.

C1, C2 🡪 Last 32 byte.

At the beginning we establish the connection as shown:



Here is the message we will attack to get the last 6 bytes from it.

Now we are going to talk about the attacking algorithm:

1. We will create an array of byte array of D2 and CC1.
   1. D2 is Static, so his values never changed after examined.
   2. CC1 is dynamic, so its value will change on each iteration.
2. After that we will give the last element of CC1, which is CC1[15] the all possible values from 0 to 255, while this we will do the following:
   1. Oracle.decrypt(IV + CC1 + C2)
   2. Check if the status is valid or not.
   3. If valid status then here we find value that will helps us.
3. The found value will be used as follows:
   1. the static D2[16 – K] value 🡪 D2[16-K] = CC1[16-K] xor K
   2. K is incremented
   3. All the CC1 values from CC1[15] to current 16-K value will be treated as follows:
      1. CC[i] = D[i] xor K
4. We will do the 3rd step 6 times until we get the last 6 bytes.

These are the valid status results and the CC1 content each K-time:



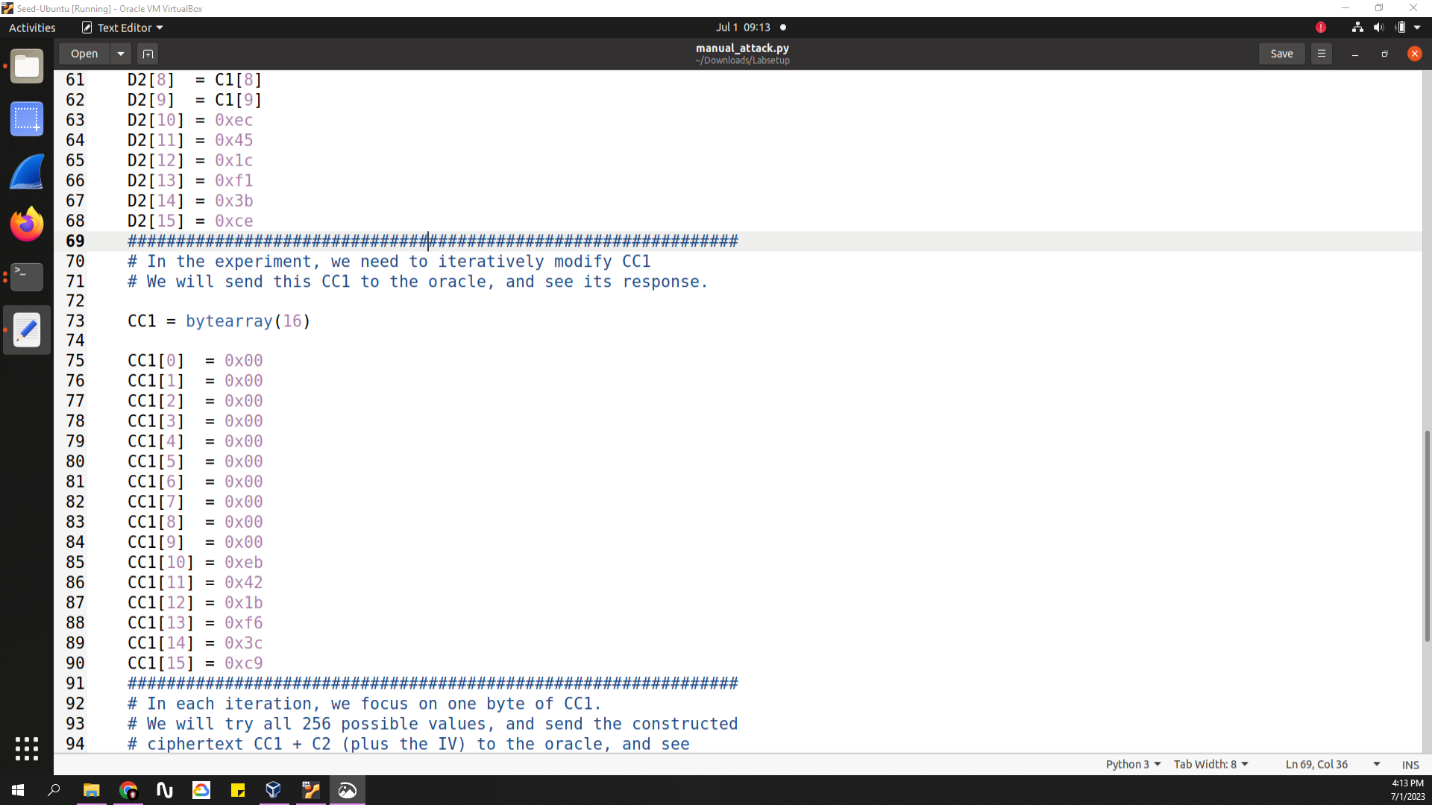
Look above on the figure:

The red rectangle represents the results when K = 1.

The yellow rectangle represents the results when K = 2.

You can notice that the CC1 values are dynamic and changes each K- time.

Now our results from applying the above algorithm is:



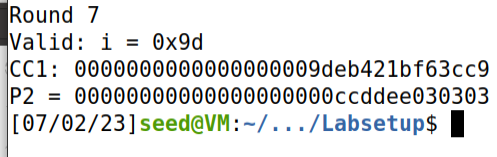
As shown in the figure, the last 6 bytes of:

D2: 0xec451cf13bce

CC1: 0xeb421bf63cc9

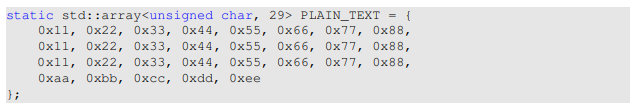
And K = 7

After that we xor D2 with C1 to get P2



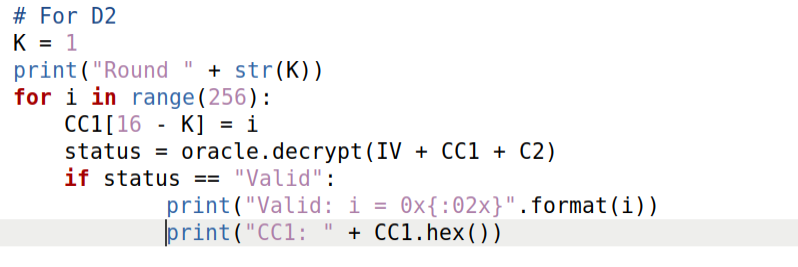
As shown the P2: 0x00000000000ccddee030303

Removing the padding 🡪 P2: 0x00...00ccddee.

By comparing it with this   
we know that our work is correct!

CODE:

The same of manual\_attack.py, but we modify the for loop as follows:



**Task 3:** Padding oracle attack Lvl2

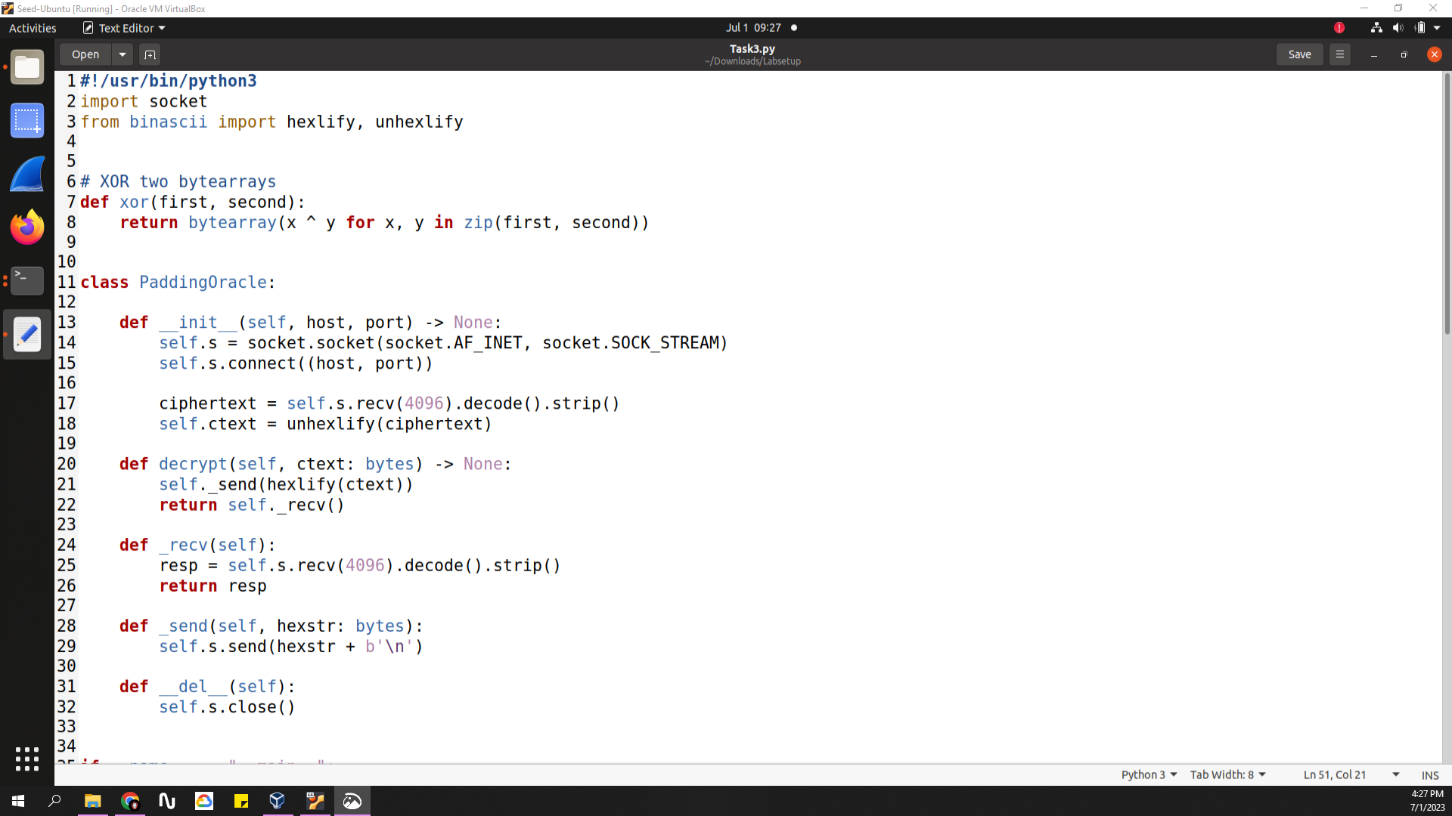
This task is the same as task2, so we follow the same algorithm followed in task 2.

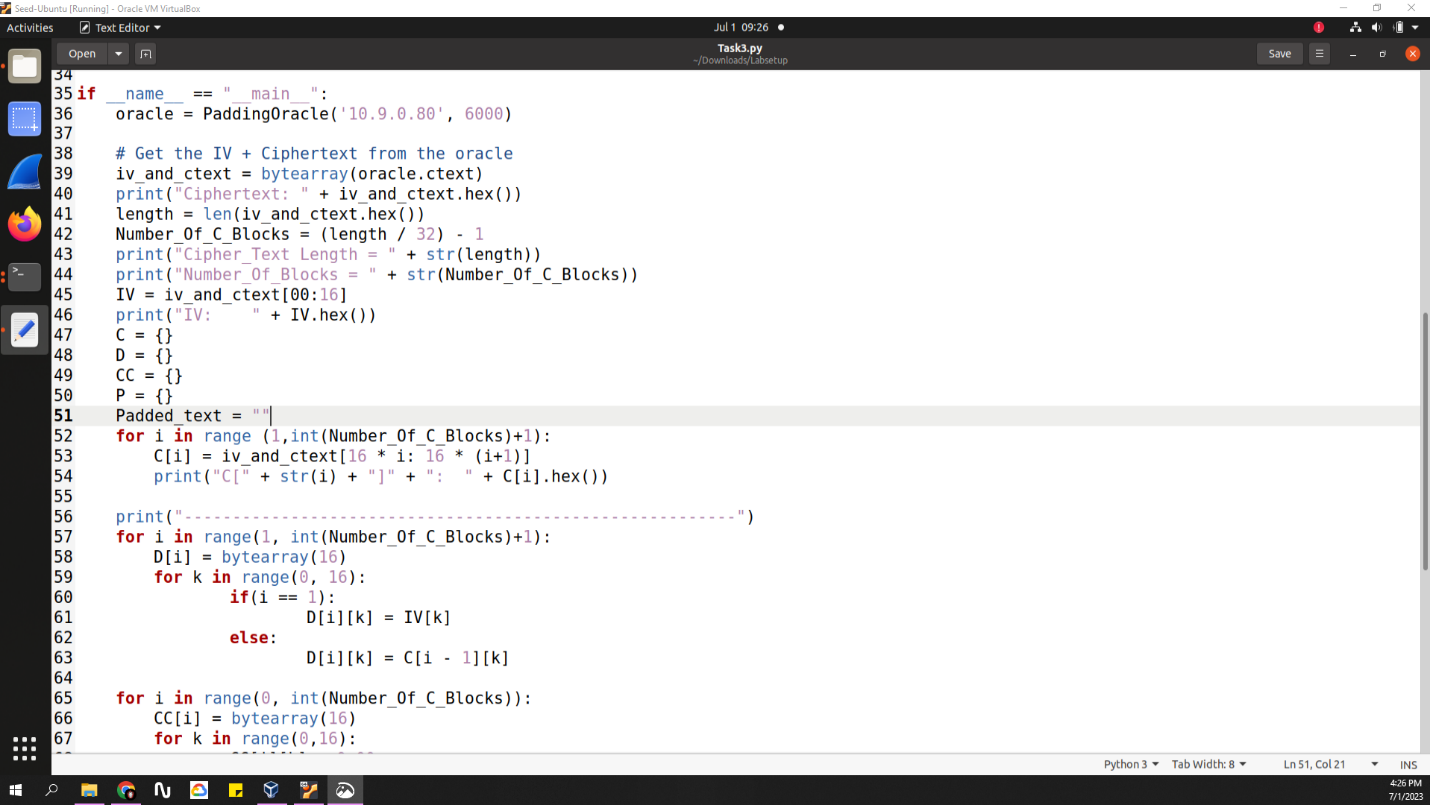
But here we take port 6000, this port hidden message is 64 byte, the first 16 for IV, and the last 48 are divided into 3 a6 byte blocks.

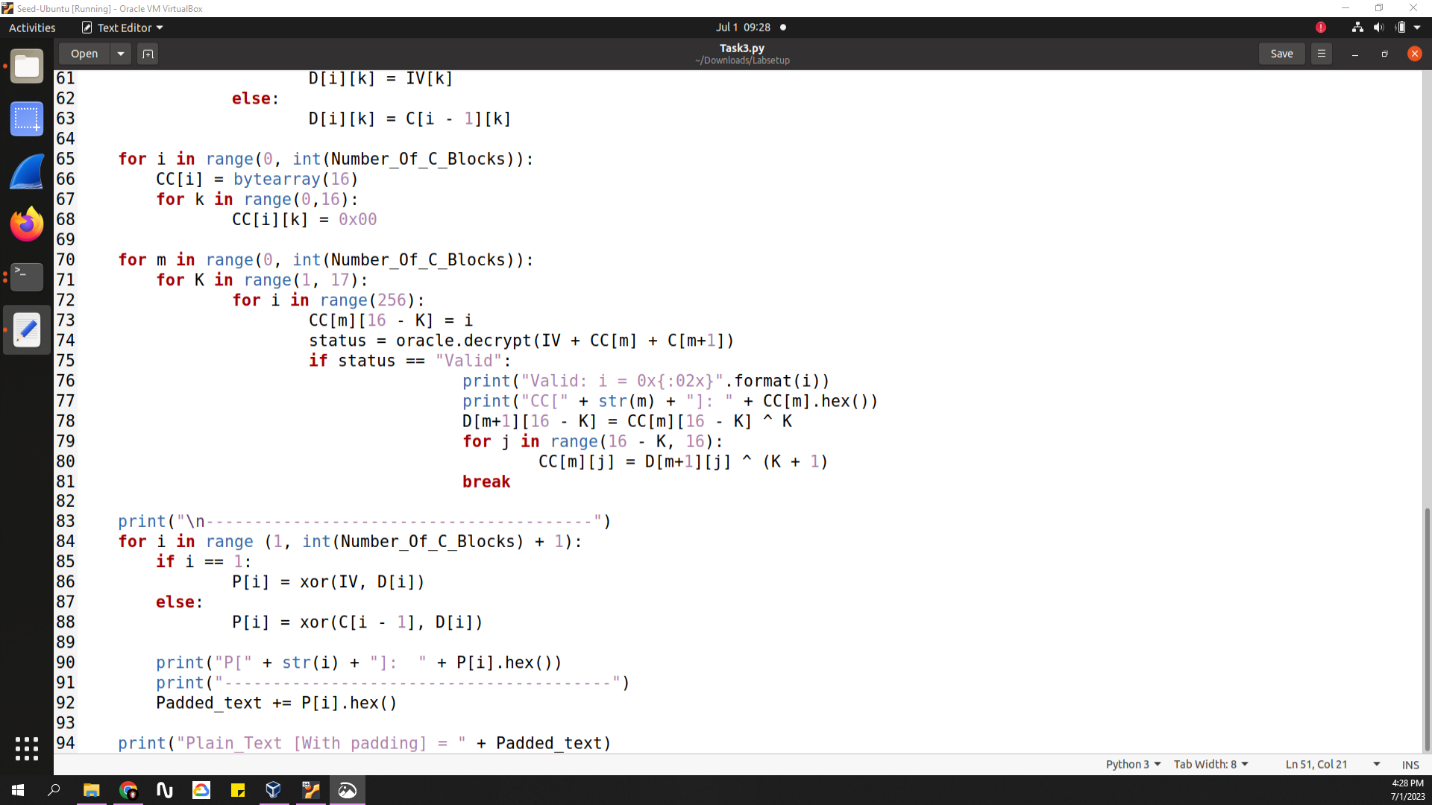
The task here requires the attack on the whole message to get the hidden text behind it.

The code is saved in Task3.py

CODE:

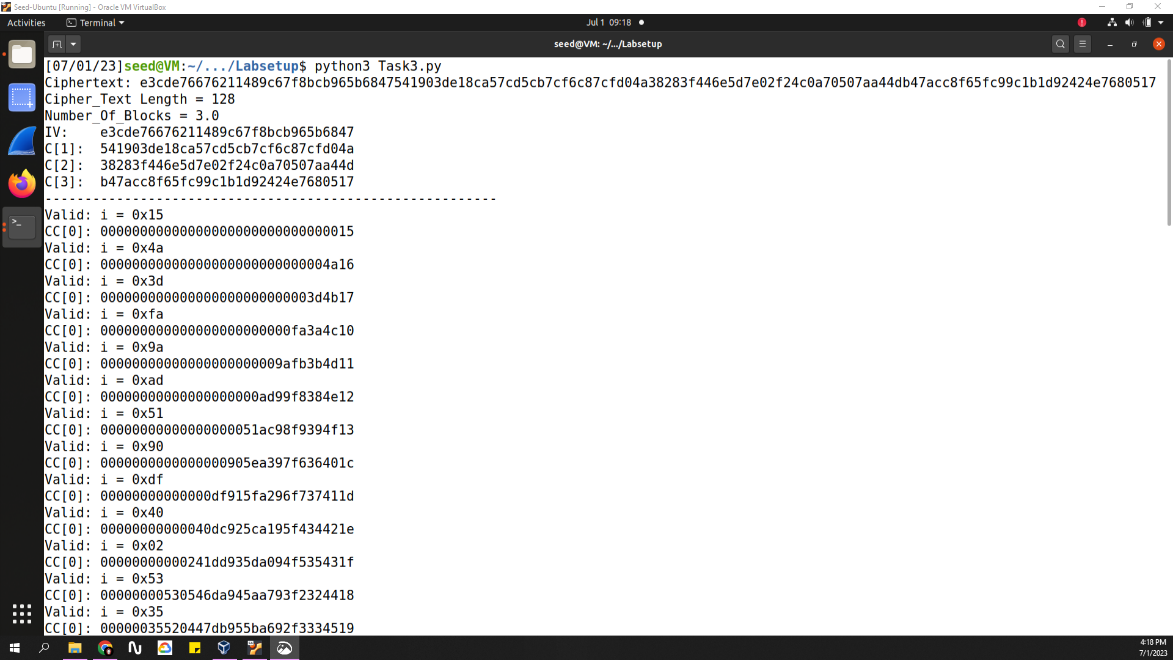


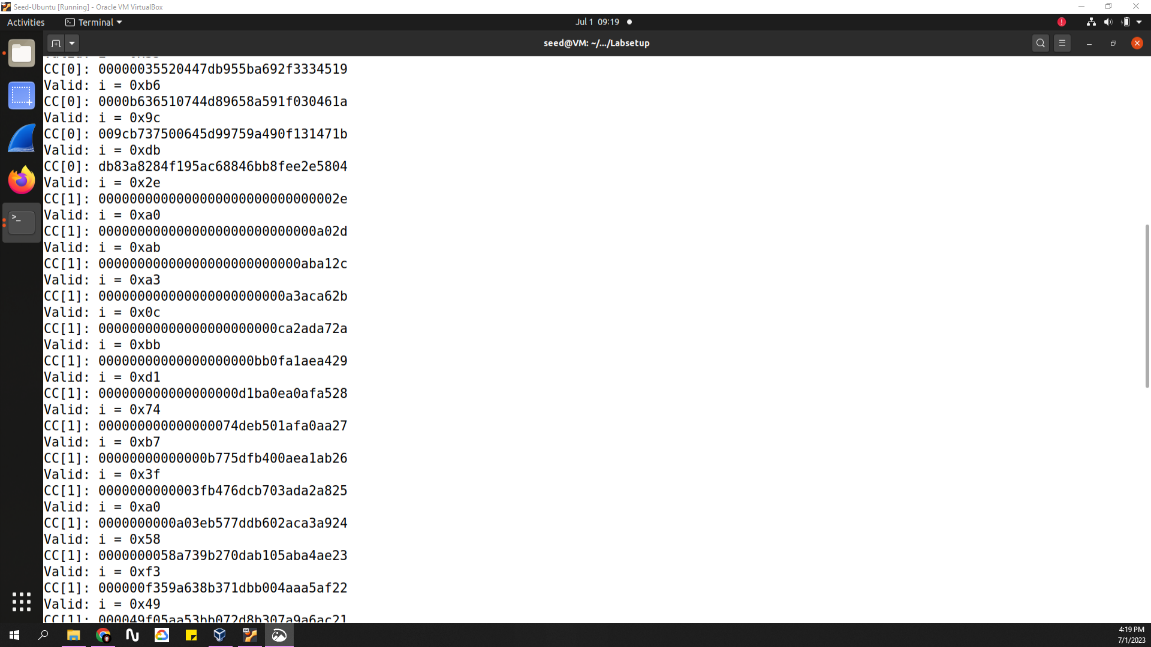


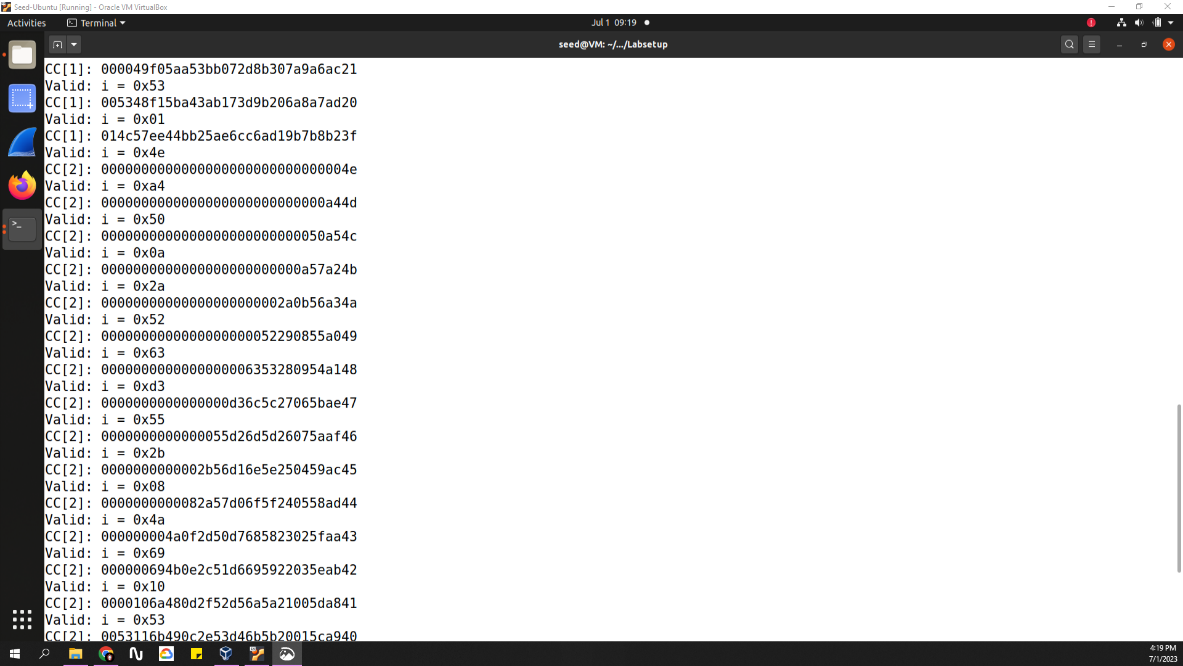


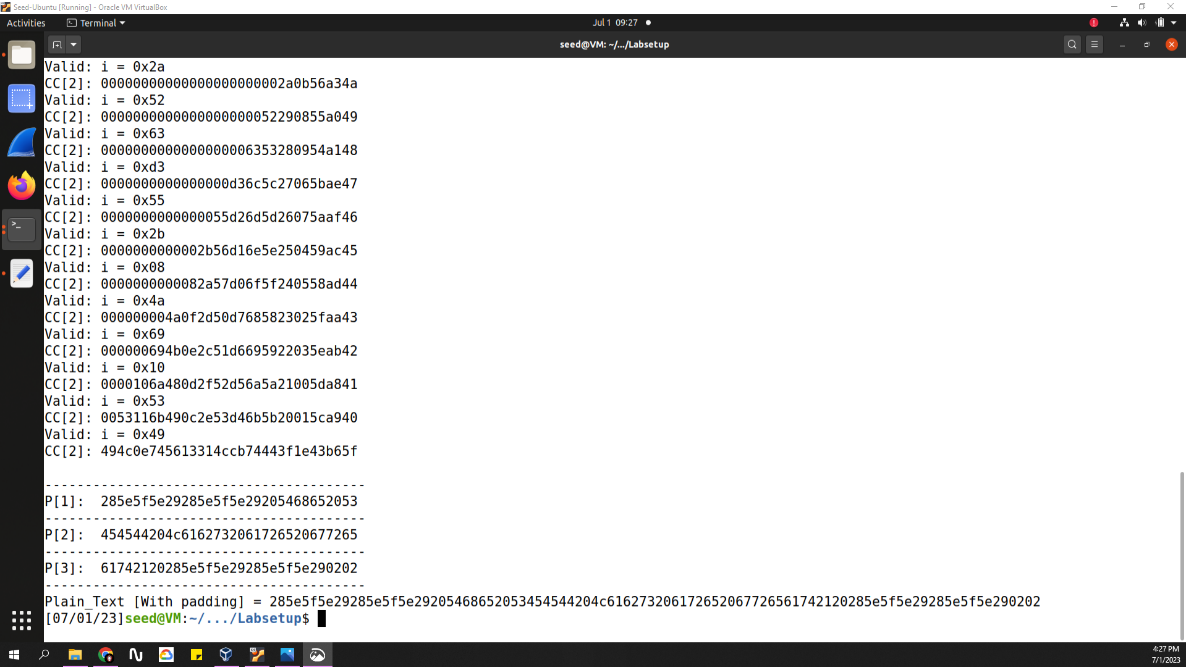
The code here is strong code that will accept any cipher-text message of any length and do the decryption/ oracle attack on it to get the final message

The terminal output is:









As shown the code bring the message, print it, then print the number of words, and the number of C blocks required for the attack.

Here the output shows IV, C[1], C[2], C[3].

The attack starts on C1 then C2 then C3, after getting all D's we xor D with previous C to get the plain text.

As shown P1, P2, P3 then we combine them to print the whole Plain-text message.

Conclusion

Conclusion

To conclude, in this lab, we explored the Cryptography Padding Oracle Attack to uncover hidden messages. The tasks involved understanding AES-128-CBC encryption and decryption, and then applying the padding oracle attack to retrieve secret messages from different ports. By exploiting vulnerabilities in the padding scheme, we iteratively decrypted ciphertext blocks to extract the plaintext.