CS582 Lab1- Padding Oracle Lab

Jevin Modi

February 8, 2024

(i) What is the padding oracle attack?

Padding Oracle attack is a form of cryptographic attack in which the attacker tries to obtain the plain text from the cipher text through submitting queries to the oracle. This is a form of a Chosen Ciphertext Attack (CCA). The attacker submits a ciphertext to the oracle and oracle responds with a yes if the the ciphertext has a valid padding and no otherwise. There are no limits on the number of ciphertexts that the attacker can send to the oracle.

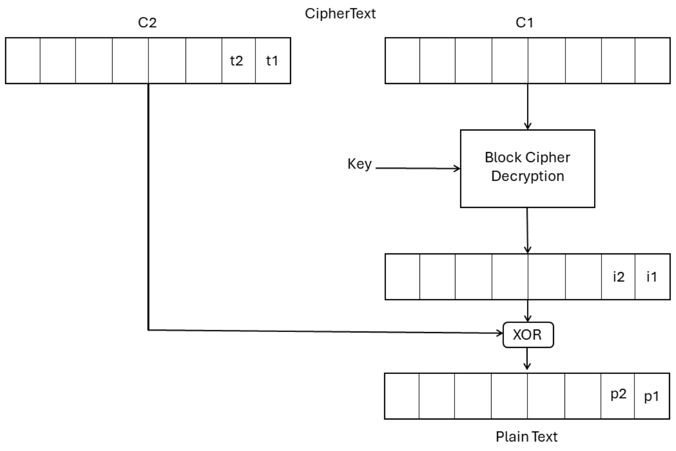


Figure 1: Padding Oracle attack

For this example, lets consider the size of a block to be 8 characters. We have a ciphertext of length 16, which is divided 2 blocks each of length 8 (C1 and C2). C1 is the current block we wish the recover and c2 is the previous block. We wish to recover the plain text from the given ciphertext. We have access to oracle. we can submit ciphertext to the oracle and it will decrypt it and return true if the plain text has valid padding and false otherwise. The oracle will just return the boolean value. This is all the information we have now. The entire attack is based on the fact that

$$a \oplus b = c \tag{1}$$

and subsequently

$$b \oplus c = a \tag{2}$$

and

$$a \oplus c = b \tag{3}$$

To recover the last character of the plain text from C1, we create a copy of C2 and assign it some random values. In my case I have assigned `\x70` to the entire C2 block. Now our goal is to modify t1 such that when the oracle get the plain text, p1 would be set to `\x01` and hence the oracle will return true as it will be a valid padding. We have 0-255 values that we can set t1 to. At each value we send the query to the oracle until we get True. As soon as we get true, We can determine the value of i1. since, we know t1, we can get i1 through the following equation:

$$t1 \oplus 01 = i1 \tag{4}$$

Now we know i1 and we know the original t1, we can recover the plain text p1 through the following equation:

$$t1 \oplus i1 = p1 \tag{5}$$

Now to recover p2, we need to set t1 and t2 in the ciphertext so that after decryption, the oracle will get `\x02` in p1 and p2. Since we already know i1, we set t1 in the following way

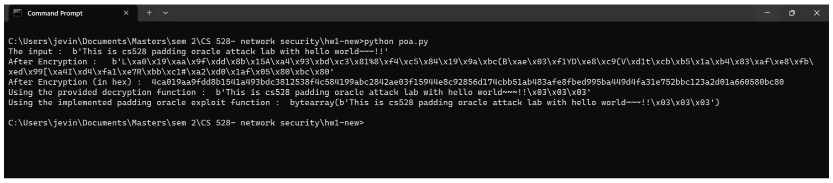
$$i1 \oplus 02 = t1 \tag{6}$$

To set t2, we try 0-255 values until the oracle returns True. After we get true, we can recover i2 the same way discussed above and subsequently recover p2. We can do this until we recover the entire plain text. Since we are given an initialization vector too, While recovering the first block of encryption, we modify the Iv and send queries to the oracle. Technically speaking, the IV is the previous block of the first block.

(iii) Description of your implementation with screenshot(s).

Output:

The output after running the python file is given in the image below.



```
C:\Users\jevin\Documents\Masters\sem 2\CS 528- network security\hw1-new>python poa.py
The input :  b'This is cs528 padding oracle attack lab with hello world~~~!!'
After Encryption :   b'L\xa0\x19\xaa\x9f\xdd\x8b\x15A\xa4\x93\xbd\xc3\x81%8\xf4\xc5\x84\x19\x9a\xbc(B\xae\x03\xf1YD\xe8\xc9(V\xd1t\xcb\xb5\x1a\xb4\x83\xaf\xe8\xfb\
xed\x99[\xa4I\xd4\xfa1\xe7R\xbb\xc1#\xa2\xd0\x1af\x05\x80\xbc\x80'
After Encryption (in hex) :  4ca019aa9fdd8b1541a493bdc3812538f4c584199abc2842ae03f15944e8c92856d174cbb51ab483afe8fbed995ba449d4fa31e752bbc123a2d01a660580bc80
Using the provided decryption function :  b'This is cs528 padding oracle attack lab with hello world~~~!!\x03\x03\x03'
Using the implemented padding oracle exploit function :  bytearray(b'This is cs528 padding oracle attack lab with hello world~~~!!\x03\x03\x03')

C:\Users\jevin\Documents\Masters\sem 2\CS 528- network security\hw1-new>
```

Figure 2: Padding Oracle Attack Output

Description of Implementation:

The code has been divided into two parts- 1) a for() loop that goes from last block till the second block and 2) first block. The reason for dividing it is because there is no previous block to the first block and hence the guesses need to be made in the IV whereas for all the other blocks there is a previous block and hence the guesses are made in the previous block and the IV remains the same. This is the only difference between the two blocks of code implementation. I will briefly describe the working in the paragraph below.

The `for()` loop goes from the last block to the second block. In the first iteration the last block is the `current_block` and the second last block is the `prev_block`. In the next iteration the second last block is the `current_block` and the third last block is `prev_block` and so on. The `block_ans_inter` is created to store the i (from Figure 1) values. I have created `iv_new` with values `\x70`. This could be anything greater than `\x10`. I have done this to specifically mitigate the case where after decryption the plain text is in the form (for example) -
`This is cs528 padding oracle attack lab with hello world~~!!\x03\x03\x03`
So if we do not use `iv_new` with random values, then it might be difficult for us to get a true (valid padding) from the oracle as the last character of decrypted plain text to \x01 and the second last and third last characters are \x03\x03. Effectivley giving us \x03\x03\x01 which is an invalid padding.

We go over each character of the block from the last to the start. We set the values of `iv_new` so that we get the required padding in plain text when the oracle decrypts it. This is possible since we know the i (from figure 1) values. For the index that we are trying to decrypt, we make guesses of 0-255 values, join the ciphertext and send it to the oracle along with the IV. The instance at which the oracle returns True, we know that it is a valid padding. we then update the i value for that character. We get the plain text by

$$i_{\_} \oplus prev\_block[\_] = plain\_text[\_] \qquad (7)$$

The " _ " here represents the index.
For the first block we do the same thing but we make guesses in the IV. So while sending queries to the oracle we essentially have the same ciphertext but modified IV for each character.

This is the description of the implementation of the padding oracle attack.