Applied Cryptography Lab-10 Manual

02 November 2022 01:02

Prerequisites

Labsetup files - https://seedsecuritylabs.org/Labs_20.04/Crypto/Crypto_Padding_Oracle/

Task 1: Getting Familiar with Padding

For some block ciphers, when the size of a plaintext is not a multiple of the block size, padding may be required. The PKCS#5 padding scheme is widely used by many block ciphers. This task involves understanding this padding scheme.

Step 1: Create a file with five characters

Command

```
$ python3 -c "print('A'*5)" > P
```

Step 2: Encrypt, then decrypt this file using AES 128 CBC mode

Commands

```
$ openssl enc -aes-128-cbc -e -in P -out C
$ openssl enc -aes-128-cbc -d -nopad -in C -out P_new
```

Step 3: View the decrypted file using xxd

```
$ xxd P_new
```

Note your observations and repeat steps 1 through 3 for files with length 10 and 27 characters.

Question:

What do you deduce about the encryption scheme?

Setup for tasks 2 and 3

1. Download the `Labsetup.zip` file from the seedlabs website and unzip it to your working directory
2. Navigate to the Labsetup folder in your terminal and run the command `docker-compose up`

This should have started an oracle server on the ip `10.9.0.80`, which should be accessible via netcat

Commands

```
$ nc 10.9.0.80 5000 # level 1
$ nc 10.9.0.80 6000 # level 2
```

A python script, manual_attack.py, is also provided in the folder. Create a copy of this, call it automated_attack.py

Task 2: Padding Oracle Attack (Level 1)

On connecting to the oracle, you are provided with an encrypted message. This message has been encrypted with AES 128, CBC mode, and follows the PKCS#5 padding scheme.
You might note that every time you connect to the server, the encrypted message is exactly the same. This means keys and IV are not generated every time the server is called.

The script provided, manual_attack.py, connects to the server and bruteforces through the 255 possible values for the 16th byte of the first ciphertext block, such that the last byte of the plaintext generated gives '01', which is a valid pad.

The array D2 is supposed to hold the output of the AES encryption block 2. C1 and C2 are the ciphertext blocks and CC1 is a copy of C1 that you're supposed to alter in order to derive the contents of D2 by performing the Padding Oracle Attack. This is done as follows:

Step 1: Run the script

Command

```
$ ./manual_attack
```

Step 2: Get the value of i from the output, and use it to find the values of CC1 and D2

Example

K = 1	K = 2
D2[15] = 0x01 ^ i	D2[14] = 0x02 ^ I
CC1[15] = D2[15] ^ 2	CC1[15] = D2[15] ^ 3 | CC[14] = D2[14] ^ 3

Step 3: Update the values of CC1 and D2, then increment the value of K by 1

Repeat steps 1 through 3 till K = 16

In the end, the decrypted plaintext is printed out by the script. Cross-check the answer, it should be
11223344556677881122334455667788112233445566778 8aabbccddee030303

Task 3: Padding Oracle Attack (Level 2)

This task is similar to the previous task, except this time the server generates a different encryption key and IV every time you connect to it. Therefore, getting to the plaintext manually isn't an option here.

Firstly, open the copy of manual_attack.py that we created prior to starting Task 2, and change the port on **line 34** to 6000 from 5000.

Next, please attempt to change this script to automate all the steps from Task 2.
(The gist of the code is provided below, but try doing it on your own first as a challenge for yourself!)

Code

```
for K in range(1,17):
    for i in range(256):
        CC1[16 - K] = i
        P2 = xor(CC1, D2)
        status = oracle.decrypt(IV + CC1 + C2)
        if status == "Valid":
            print("Valid: i = 0x{:02x}".format(i))
            print("CC1: " + CC1.hex())
            D2[16-K] = K ^ i#C1[16-K]
            for j in range(16-K,16):
                CC1[j] = (D2[j] ^ K+1)
            break
```

Provide screenshots of your code and output