Huzaifah Majid
Computer Security
Professor Leo Fan
March 10 2023

Padding Oracle Lab

Introduction

The purpose of this lab is to introduce a class of cryptographic attacks called padding oracle attacks. Specifically, I am tasked with exploiting a vulnerability in a web application that uses CBC encryption with PKCS#7 padding.

In this lab I will learn how the padding oracle attack works and how it can be used to decrypt ciphertext without the attacker knowing the encryption key. I'll also learn how to use tools like Burp Suite and Python to automate the process of sending crafted ciphertext to a server and parsing the response.

Additionally, I'll gain experience building virtual environments, configuring web applications, and using various tools to carry out attacks. After completing this lab, I will have a better understanding of the security risks associated with cryptographic protocols and how to mitigate them.

Background

A padding oracle attack is a type of cryptographic attack that targets systems that use padding block ciphers. Padding is used to ensure that the plaintext message is a multiple of the pre-encryption block size. In CBC mode, the ciphertext of each block depends on the previous block and the encryption key. A padding oracle attack allows an attacker to decrypt the ciphertext by sending specially crafted ciphertext to the server using the padding verification process and parsing the response. An attacker can use this information to decrypt the entire message. This attack works by exploiting the fact that the server can reveal information about the effectiveness of the padding when decrypting the ciphertext. If the padding is correct the server will return a success message, but if the padding is incorrect the server will return an error her message. By repeatedly sending crafted ciphertext to the server and analyzing the responses, an attacker can obtain information about the plaintext message and eventually decrypt it. Padding oracle attacks are a serious threat to systems using block ciphers with padding and can compromise the confidentiality of sensitive data if not properly mitigated.

Procedure

<u>File 1 creation:</u>

```
$ echo -n "12345" > P1
$ openssl enc -aes-128-cbc -e -in P1 -out C1
$ openssl enc -aes-128-cbc -d -nopad -in C1 -out P1_new
$ xxd P1_new
```

The output shows the padding bytes added to the file, which are 0b repeated 11 times to make a total of 16 bytes (one full block) after padding.

File 2 creation:

```
echo -n "1234567890" > P2
openssl enc -aes-128-cbc -e -in P2 -out C2
openssl enc -aes-128-cbc -d -nopad -in C2 -out P2_new
xxd P2_new
```

The output shows the padding bytes added to the file, which are 06 repeated 6 times to make a total of 16 bytes after padding.

File 3 creation:

```
echo -n "1234567890123456" > P3
openssl enc -aes-128-cbc -e -in P3 -out C3
openssl enc -aes-128-cbc -d -nopad -in C3 -out P3_new
xxd P3_new
```

The output should show the padding bytes added to the file, which should be 10 repeated 16 times to make a total of 32 bytes (two full blocks) after padding.

4.1

To perform the padding oracle attack understanding that the padding works in the PKCS#5 scheme in crucial. In this scheme, if the length of the plaintext is not a multiple of the block size (16 bytes for AES-CBC), padding bytes are added to the end of the plaintext to make its length a multiple of the block size. The padding bytes have a value equal to the number of padding bytes added, so if 3 bytes are added, each of them will have a value of 0x03.

In the attack, I will modify the last byte of the previous block of the ciphertext and observe whether the padding is valid or not. If the padding is valid, the last byte of the plaintext block is equal to the XOR of the last byte of the previous ciphertext block and the padding value. I then repeat this process for the second-to-last byte of the plaintext block, and so on, until I have recovered the entire block.

To perform the attack:

1. Connect to the oracle using "nc 10.9.0.80 5000".
2. Send the IV concatenated with the ciphertext to the oracle to get the ciphertext decrypted.
3. Modify the last byte of the previous ciphertext block and send the modified ciphertext to the oracle.
4. If the padding is valid, the oracle will return "Good padding". If the padding is invalid, the oracle will return "Bad padding".
5. Repeat steps 3 and 4 for each byte in the block until we have recovered the entire block.

4.2

To derive the plaintext manually, I would need to use a padding oracle attack to figure out the values of D2. I can do this by iterating through each byte of D2 and constructing CC1 to send to the oracle to see which value of CC1 makes the padding valid. I will use python since I am most comfortable in that language

1. Get the ciphertext from the oracle and separate it into two blocks, C1 and C2.
   ```
   oracle = PaddingOracle('10.9.0.80', 5000)
   iv_and_ctext = bytearray(oracle.ctext)
   IV = iv_and_ctext[0:16]
   C1 = iv_and_ctext[16:32]
   C2 = iv_and_ctext[32:48]
   ```

2. Initialize arrays
   ```
   D2 = bytearray(16)
   CC1 = bytearray(16)
   ```

3. Iterate though each byte of CC1 and send it to oracle to see which value of CC1 makes valid padding.
   ```
   K = 1
   for i in range(256):
     CC1[16 - K] = i
     status = oracle.decrypt(IV + CC1 + C2)
     if status == "Valid":
       print("Valid: i = 0x{:02x}".format(i))
       D2[16 - K] = C1[16 - K] ^ i
       print("D2: " + D2.hex())
       print("CC1: " + CC1.hex())
   ```

4. Repeat for all bytes in D2, while updating value of CC1
   ```
   for K in range(2, 17):
     for i in range(256):
       CC1[16 - K] = i ^ D2[16 - K + 1]
       status = oracle.decrypt(IV + CC1 + C2)
       if status == "Valid":
         print("Valid: i = 0x{:02x}".format(i))
         D2[16 - K] = C1[16 - K] ^ i ^ (K - 1)
         print("D2: " + D2.hex())
         print("CC1: " + CC1.hex())
   ```

5. Xor C1 with D2 to get plaintext
   ```
   P2 = bytearray([C1[i] ^ D2[i] for i in range(16)])
   print("P2: " + P2.hex())
   ```

Conclusion

In this lab I learned about padding oracle attacks. This is a type of cryptographic attack that exploits a padding validation vulnerability in certain cryptographic modes. I started by manually executing the attack against the vulnerable web application. This helped me understand the inner workings of the attack.

Next, I automated the attack process by writing a Python script to carry out the attack. This script allowed us to retrieve the secret message block by block, demonstrating the effectiveness of the padding oracle attack.

In Exercise 4.1 we learned about the CBC-MAC structure, which is used to provide data integrity in certain applications. It is possible to create fake messages that pass the CBC-MAC verification process. Which is why systems should use authentication encryption modes.

Finally, in Task 4.2 I learned about the GCM authenticated encryption mode, which provides both confidentiality and integrity of data. I was able to create a fake message that passed the GCM authentication process. This demonstrates the importance of carefully choosing and correctly implementing encryption modes.

Overall, this lab provided valuable hands-on experience with cryptographic attacks and authenticated encryption modes, helping me to better understand the strengths and weaknesses.