BIRZEIT UNIVERSITY

Faculty of Information Technology

Computer Systems Engineering Department

INFORMATION SECURITY AND COMPUTER NETWORK LABORATORY

ENCS5121

Padding Oracle Attack Lab

Task3

Prepared by: Dana Bornata – 1200284

Instructor: Dr. Ibrahim Nemer

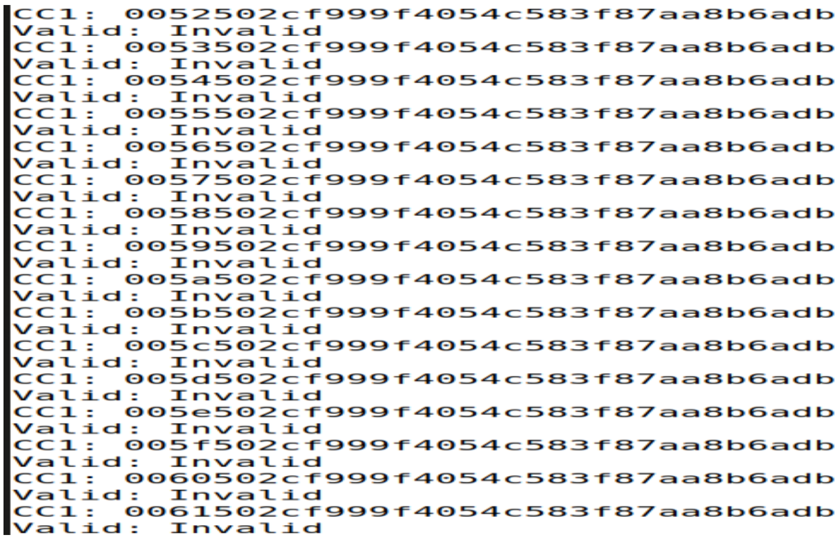
Teacher Assistant: Mohammed Balawi

Section: 1

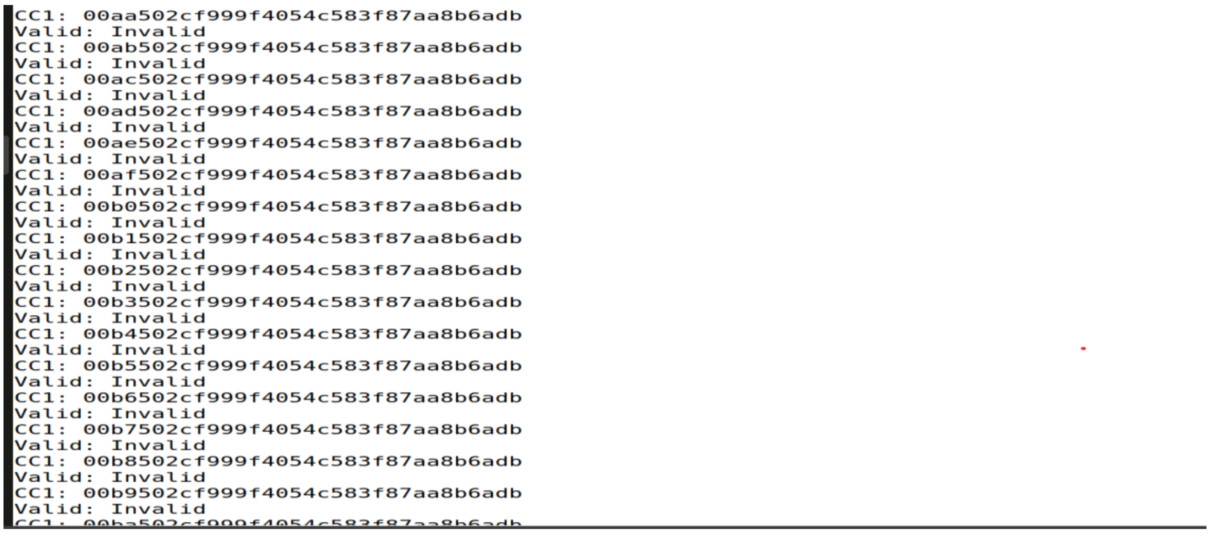
Date: 14/3/2024

In this task, the attack process will be automated, and this time, the goal is to obtain all the blocks of plaintext. When the system is started, two padding oracle servers will be launched – one for the Level-1 task and another for the Level-2 task. The Level-2 server listens on port 6000. A "Padding Oracle Attack" is a method of exploiting the validation of padding in a cryptographic message to decrypt the ciphertext. The purpose of the Python script provided in the appendix is to carry out a padding oracle attack. This script takes a ciphertext from an oracle and attempts to decrypt it without requiring the decryption key.

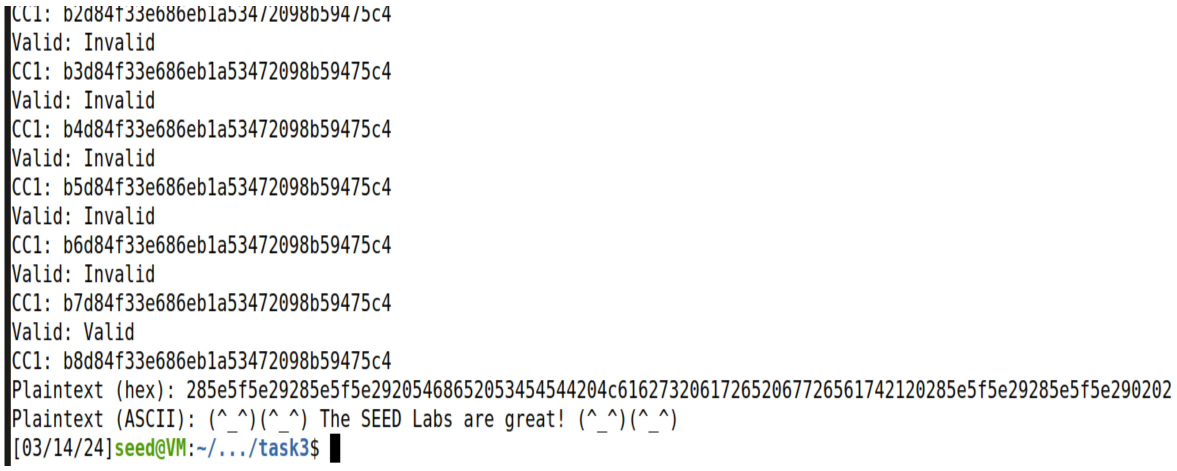
Some of output:



```
CC1:  0052502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  0053502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  0054502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  0055502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  0056502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  0057502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  0058502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  0059502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  005a502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  005b502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  005c502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  005d502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  005e502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  005f502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  0060502cf999f4054c583f87aa8b6adb
Valid:  Invalid
CC1:  0061502cf999f4054c583f87aa8b6adb
Valid:  Invalid
```



```
CC1: 00aa502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00ab502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00ac502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00ad502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00ae502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00af502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00b0502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00b1502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00b2502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00b3502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00b4502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00b5502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00b6502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00b7502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00b8502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00b9502cf999f4054c583f87aa8b6adb
Valid: Invalid
CC1: 00ba502cf999f4054c583f87aa8b6adb
```



```
CC1: b2d84f33e686eb1a53472098b59475c4
Valid: Invalid
CC1: b3d84f33e686eb1a53472098b59475c4
Valid: Invalid
CC1: b4d84f33e686eb1a53472098b59475c4
Valid: Invalid
CC1: b5d84f33e686eb1a53472098b59475c4
Valid: Invalid
CC1: b6d84f33e686eb1a53472098b59475c4
Valid: Invalid
CC1: b7d84f33e686eb1a53472098b59475c4
Valid: Valid
CC1: b8d84f33e686eb1a53472098b59475c4
Plaintext (hex): 285e5f5e29285e5f5e292054686520534545442204c616273206172652067726561742120285e5f5e29285e5f5e290202
Plaintext (ASCII): (^_^)(^_^) The SEED Labs are great! (^_^)(^_^)
[03/14/24]seed@VM:~/.../task3$ 
```

The script begins by breaking down the received ciphertext into 16-byte blocks. It then proceeds to decrypt each byte within these blocks, excluding the initial block used as an Initialization Vector (IV). Using the PKCS7 padding scheme, the decryption process starts with the last byte in the block. The script systematically tests all possible values for each byte, ranging from 0 to 255, and checks if a particular value results in the correct padding through the oracle's confirmation. When verification succeeds, the script records this byte's value before moving on to the next byte. Once the entire block is decrypted, the script incorporates these results into the final plaintext and continues with the next block.

Plaintext (hex):
285e5f5e29285e5f5e29205468652053454544204c616273206172652067726561742120285e5f5e29285e5f5e290202

Plaintext (ASCII): (^_^)(^_^) The SEED Labs are great! (^_^)(^_^)

- The code was attached to the reply to the message along with the Word file

The code:

```
import socket

from binascii import hexlify, unhexlify

# XOR two bytearrays

def xor(first, second):

  return bytearray(x^y for x,y in zip(first, second))

class PaddingOracle:

  def __init__(self, host, port):

    self.s = socket.socket(socket.AF_INET, socket.SOCK_STREAM)

    self.s.connect((host, port))

    ciphertext = self.s.recv(4096).decode().strip()

    self.ctext = unhexlify(ciphertext)

  def decrypt(self, ctext: bytes) -> str:

    self._send(hexlify(ctext))

    return self._recv()

  def _recv(self):

    resp = self.s.recv(4096).decode().strip()

    return resp

  def _send(self, hexstr: bytes):

    self.s.send(hexstr + b'\n')

  def __del__(self):

    self.s.close()

def padding_oracle_attack(oracle, IV, ciphertext):

  block_size = 16

  num_blocks = len(ciphertext) // block_size

  plaintext = bytearray()

  for block_num in range(num_blocks):

    block_start = block_num * block_size

    block_end = (block_num + 1) * block_size

    C1 = IV if block_num == 0 else ciphertext[block_start - block_size:block_start]

    C2 = ciphertext[block_start:block_end]

    D2 = bytearray(block_size)

    P2 = bytearray(block_size)
```

```
        for i in range(block_size):
            # Construct CC1
            CC1 = bytearray(block_size)
            for j in range(i + 1):
                CC1[-j - 1] = D2[-j - 1] ^ (i + 1)
            for guess in range(256):
                CC1[-i - 1] = guess
                status = oracle.decrypt(IV + CC1 + C2)
                print("Valid:", status)
                print("CC1:", CC1.hex())
                if status == "Valid":
                    D2[-i - 1] = guess ^ (i + 1)
                    P2[-i - 1] = D2[-i - 1] ^ C1[-i - 1]
                    break
        plaintext.extend(P2)
    return plaintext
def main():
    oracle = PaddingOracle('10.9.0.80', 6000)
    IV = oracle.ctext[:16]
    ciphertext = oracle.ctext[16:]
    plaintext = padding_oracle_attack(oracle, IV, ciphertext)
    print("Plaintext (hex):", plaintext.hex())
    print("Plaintext (ASCII):", plaintext.decode())
if __name__ == "__main__":
    main()
```