CS6111: Homework set 3

Soham Tripathy CS20B073

Problem 1

Padding Oracle attack
When we look at the binary code, we realize that the server decrypts the provided cookie and returns whether the decrypted message is correctly padded or not. This can be used to exploit the server since it is vulnerable to padding oracle attack. We can infer the plaintext from the give code.

```
# %%
if isvalidpad(cookie2decoded):
    d=json.loads(unpad(cookie2decoded))
    print("rollnumber: " + d["roll"], flush=True)
    print("Admin? " + d["is_admin"], flush=True)
    exptime=time.strptime(d["expires"],"%Y-%m-%d")
    if exptime > time.localtime():
        print("Cookie is not expired", flush=True)
    else:
        print("Cookie is expired", flush=True)
    if d["is_admin"]=="true" and exptime > time.localtime():
        flag = open("flags/" + str(d["roll"]) + ".txt", "r").read().strip()
        print("The flag is: " + flag, flush=True)
else:
```

From the above picture it is clear to us that during *json.loads* we actually need a json object to be returned from the unpad function. Thus the plaintext we want to achieve is the paded version of the following.

$$\{"roll" : "CS20B073", "is\_admin" : "true", "expires" : "2023-12-31"\}$$

The padded version of the above will contain 80 bytes of data, thus having 5 blocks. We will work on each block trying to predict a IV, for which we would get the desired plaintext.

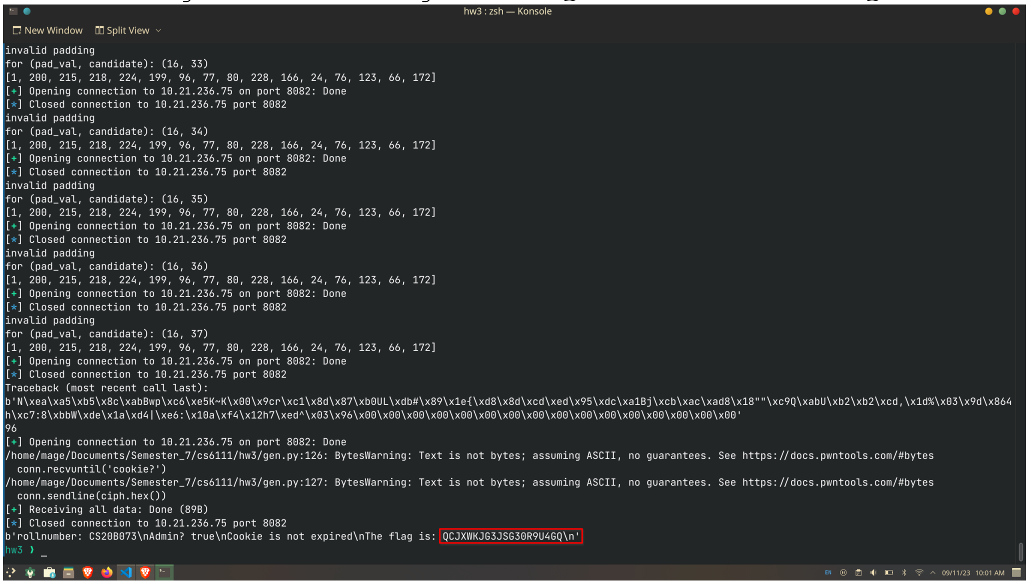
Padding Oracle Attack
Consider a random IV of 16B, some cipher text C of 16B and correctly padded plain text to be P. We know that during decryption of C, we xor IV with $Dec(C, k)$ and get the padded text P′, this is then checked for validation. Now, we can change the last byte of IV, to change the last byte of P′ and hence creating a new plaintext P″ for which the validation holds true and we know that the last byte of this P″ would be $0x01$. By taking xor of $0x01$ with last byte of IV, we get IV_2 which zeros the last byte of the plaintext it generates. We would continue this for all the 16B to generate $IV_{zeroing}$ which would produce plaintext with all zeros.

The Attack
We consider two random 16B strings, one is the IV and the other is some fixed string. We would change the IV according to the padding oracle attack, to get the zeroing IV. This means that for this IV, our plaintext would be all zeros. Now we would xor this zeroing IV with the desired plaintext block of 16B to get the correct IV.

$$IV_{zeroing} \oplus R = O$$
$$IV_{zeroing} \oplus DPLAIN \oplus R = DPLAIN$$

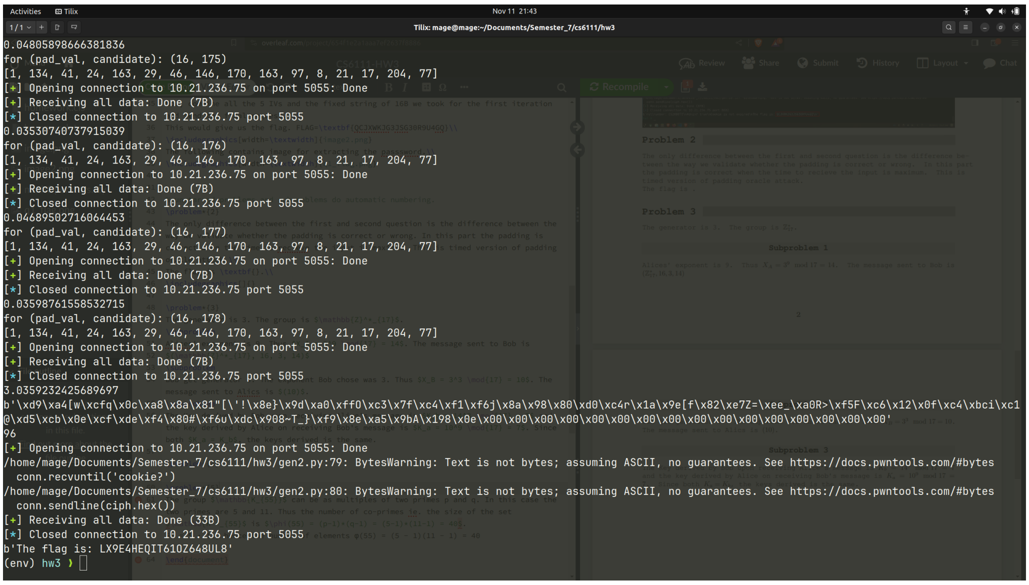
Now, the correct IV would become the fixed string and by taking another random IV_2 we would start the process again. Since, we have 5 blocks in our plain-

text blocks, we would repeat the above step for 5 times. This process would start from the last block proceed backwards till the first one.
Now, combine all the 5 IVs and the fixed string of 16B we took for the first iteration and send the final request.
This would give us the flag. FLAG=**QCJXWKJG3JSG30R9U4GQ**



Problem 2

The difference between the first and second question is in the way which is used to validate whether the padding is correct or wrong. In this part the padding is correct when the time to recieve the input is maximum. This is timed version of padding oracle attack.
The flag is **LX9E4HEQIT61OZ648UL8**.



Problem 3

The generator is 3. The group is $\mathbb{Z}_{17}^{*}$.

Subproblem 1

Alices' exponent is 9. Thus $X_A = 3^9 \mod 17 = 14$. The message sent to Bob is

$(\mathbb{Z}_{17}^*, 16, 3, 14)$

Subproblem 2

Bob get generator 3. The exponent Bob chose was 3. Thus $X_B = 3^3 \mod 17 = 10$. The message sent to Alics is (10).

Subproblem 3

The Key derived by Bob upon receiving Alice's message is $K_b = 14^3 \mod 17 = 7$ and the key derived by Alice on receiving Bob's message is $K_a = 10^9 \mod 17 = 7$. Since both $K_a = K_b$, the keys derived is the same.

Problem 4

El Gamal encryption scheme is not CCA-secure. Let $\mathcal{A}$ be an adversary with the following algorithm.

Algorithm

* Upon receiving security parameter 1^n and a public key $(\mathbb{G}, q, g, h)$ output two messages m_0 and m_1.
* Choose a random bit $b \in \{0, 1\}$, and get an encryption for m_b. Let the ciphertext for the encryption of m_b be $\langle c_0, c_1 \rangle$.
* Now take a random element $k \in \mathbb{G}$ and multiply c_1 with k to obtain c_1'.
* Query the decryption oracle for c_1'. It receives a message m'.
* Output 0 if $\frac{m'}{k} = m_0$ otherwise 1.

Correctness

The probability of the adversary to guess the correct answer is 1. We know that $\langle c_0, c_1 \rangle = \langle g^y, g^z \cdot h_b \rangle$. Now, $\langle c_0, c_1' \rangle = \langle g^y, x * c_1 \rangle = \langle g^y, x \cdot g^z \cdot h_b \rangle$. We can observe that the decryption of c_1' is $m' = x \cdot m_b$, this when divided by x gives m_b. Hence the El-gamal encryption scheme is not CCA-secure.

Problem 5

The group $\mathbb{K}_{55}$ can be as multiples of two primes p and q. In this case the two primes are 5 and 11.

Subproblem 1

The number of co-primes ie. the size of the set $\mathbb{K}_{55}$ is $\phi(55) = (p-1) * (q-1) = (5-1) * (11-1) = 40$.

Subproblem 2

$f_3(6) = [6^3 \mod 55] = 41$

Subproblem 3

We know that if $d = e^{-1} \mod |\mathbb{K}_{55}^*|$, then $f_d(g)$ is the inverse of $f_e(g)$ (from the text book). Hence, we can say f_{27} will compute the inverse of f_3 as $3 * 27 \mod 55 = 81 \mod 40 = 1$.

Subproblem 4

We know from the above subproblem that $f_3^{-1}(x) = f_{27}(x)$, thus $f_3^{-1}(2) = f_{27}(2) = 2^{27} \mod 55 = 18$.