RSA Encryption(TASK 5)

Peeter Tarvas, 202402641

I used python for the task since it's the easiest language to do this for me. In this attack we exploit the vulnerability of RSA implementation that uses a parity oracle. This means that the server leaks information whether the decrypted ciphertext is even or odd, enabling a chosen-ciphertext attack to recover a secret plain text.

Attack Surface

The server encrypts a secret message containing a sensitive string (`secret`) and stores it in a cookie. The attacker's goal is to decrypt this ciphertext by leveraging the parity oracle.

Details

The server uses RSA encryption with PKCS#1 v1.5 padding to protect messages so the key components are>
Public key: Mod N and public exponent e
Private key: Mod N and private exponent d
PCKS#1 Padding: Adds to the message so that we can ensure messages are structured securely before encryption.

/quote endpoint is a parity oracle. When a text is submitted, the server decrypts and checks if the plaintext is even or odd, this is done by two responses:

1. "**I do not like even numbers.**": Indicates the decrypted plaintext is even.
2. **Other responses**: Imply the plaintext is odd.

This behavior allows us to iteratively refine guesses about the plaintext by observing the parity (even/odd) of manipulated ciphertexts.

Attack algorithm

Fetch public key

The attacker retrieves N and e from the server by calling /pk endpoint.

Get ciphertext

Server provides an encrypted authentication token, c, as the cookie. Convert it into an integer called c_int.

Attack search(Binary search)

The attack is basically iterating and narrowing down the possible range of plaintext m by:

a. set lower L = 0 and Upper U = N
b. For each iteration compute new ciphertext c' = c * 2**e mod. Which decrypts to (2 ** k) * m mod N - doubles the plaintext each step
c. Submit the c' to server and observe the parity of the decrypted plaintext
d. m =' 2*m mod N
    i. if m is even 2m < N, so m < N / 2. Update U = (L + U) / 2
    ii. if m is odd 2m >= N, so m > N / 2. Update L = (L + U) / 2
e. Repeat: Perform this process for N iterations which is enough to get all the bits of m

Recovering and unpadding the PKCS#1 v1.5

After recovering m, the we remove the PKCS#1 v1.5 padding to extract the secret. The padding format has a structure that the message starts with 0x00 0x02, followed by random bytes and a 0x00 separator before the plaintext.

Output

The recovered secret is "Not using proper OAEP is dangerous ..."