Hướng dẫn thực hành thám mã RSA mục đích 2
labtainer cryp-attak-rsa-prime

I.Tổng quan

Bài tập này minh họa việc phá bản mã hóa RSA theo Phương thức Tấn công mục đích 2: Tấn công trong trường hợp khóa công khai của người tấn công và người bị tấn công là các số nguyên tố cùng nhau.

Bài tập này giả định rằng sinh viên có một số kinh nghiệm lập trình python cơ bản ,quen thuộc với các cấu trúc dữ liệu đơn giản và một số thuật toán, công thức thám mã. Yêu cầu sinh viên hiểu rõ về quy trình thám mã một cách mạch lạc nhất. Tuy nhiên, nếu chưa hiểu rõ về thám mã hay các vấn đề thám mã RSA thì có thể tham khảo tại giáo trình "Mật mã học cơ sở ver2" tác giả Đỗ Xuân Chợ biên soạn.

II.Môi trường thực hành

Bài thực hành này chạy trong khuôn khổ labtainer, tại http://nps.edu/web/c3o/labtainers. Trang web bao gồm các liên kết đến máy ảo được dựng sẵn có cài đặt Labtainers, tuy nhiên Labtainers có thể chạy trên bất kỳ máy chủ Linux nào hỗ trợ chứa Docker hoặc trên Docker Desktop trên PC và Mac. Từ thư mục labtainer-student của sinh viên hãy bắt đầu Labtainers bằng cách sử dụng:

- labtainer <labtainer_name>

Một liên kết đến hướng dẫn sử dụng bài thực hành sẽ được hiển thị.

Trường hợp làm lại labtainer:

- labtainer -r <labtainer_name>

III. Nhiệm vụ

Nếu một thông điệp được gửi tới nhiều người dùng có các khóa công khai e nguyên tố cùng nhau thì kẻ tấn công có thể giải mã được thông điệp mà không cần tìm khóa bí mật. Trong trường hợp này, kẻ tấn công phải tìm được một cặp khóa công khai nguyên tố cùng nhau. Do các khóa được sinh ra ngẫu nhiên bởi hệ thống nên điều này hoàn toàn có thể xảy ra.

Ý tưởng: Thông điệp M được gửi tới hai người có khóa công khai tương ứng là e1 và e2. Hai khóa này nguyên tố cùng nhau.

Trong thực tế, việc truyền bản mã hóa RSA với khóa công khai Ku(e,n) là tương đối an toàn với số n đảm bảo nhưng vẫn có khả năng thám mã được từ khóa công khai đó. Bản mã bị phá do 2 trường hợp là khóa chưa đủ mạnh hoặc các khóa được truyền đi với một số điều kiện đặc biệt có thể thám mã được. Trong thực tế việc truyền khóa công khai dưới dạng Ku(e,n) là điều ít khi xảy ra mà thay vào đó khóa công khai thường chuyển qua một dạng PEM "(Privacy-Enhanced Mail) là một định dạng tệp mã hóa dữ liệu và khóa mật mã phổ biến,

được sử dụng chủ yếu trong các hệ thống liên quan đến bảo mật như TLS/SSL. Nó là một cách để lưu trữ và truyền tải dữ liệu như khóa công khai, khóa riêng, chứng chỉ, hoặc các bản thông tin mật mã."

Bài thực hành này mô phỏng quá trình thám mã sau khi nhận được bản mã và khóa công khai. Bước đầu sinh viên cần chuyển khóa công khai từ dạng pem sang khóa công khai dạng Ku(e,n). Sau đó dùng tools để giải mã công khai theo như các bước trong giáo trình.

III.1. Nhiệm vụ 1

Sinh viên cần code một đoạn mã hoặc dùng bất kỳ phương thức nào ví dụ openssl,... để chuyển đổi mã từ dạng PEM sang khóa công khai dạng chính tắc.

Cách 1: sử dụng code.

```python
from cryptography.hazmat.primitives import serialization
from cryptography.hazmat.backends import default_backend

def load_public_key_from_pem(file_path):
    with open(file_path, 'rb') as pem_file:
        pem_data = pem_file.read()

    public_key = serialization.load_pem_public_key(pem_data, backend=default_backend())

    if hasattr(public_key, 'public_numbers'):
        public_numbers = public_key.public_numbers()
        n = public_numbers.n
        e = public_numbers.e
        return n, e
    else:
        raise ValueError("File không chứa khóa công khai hợp lệ.")

pem_file_path = "public_key1.pem"

try:
    n, e = load_public_key_from_pem(pem_file_path)
    print(f"Modulus (n): {n}")
    print(f"Exponent (e): {e}")
except Exception as ex:
    print(f"Lỗi: {ex}")
```

Cách 2: sử dụng openssl.

- openssl rsa -in public_key1.pem -pubin -text

III.2. Nhiệm vụ 2

Nhiệm vụ 2.1

Sau khi sinh viên nhận được mã công khai dạng chính tắc, sinh viên dùng mã nguồn rsa_prime.py để giải mã.

Đoạn mã còn thiếu sót ở bước giải mã, sinh viên cần code đoạn mã giải theo các bước giải mã như hình dưới đây.

```
def attack(c1, c2, e1, e2, N):

    return r1
```

Yêu cầu sinh viên đọc hiểu các bước giải và thực hiện bổ sung code vào đoạn mã.

Đầu vào: kẻ tấn công có : modul chung n; các khóa công khai e_1, e_2; các bản mã C_1, C_2.

Đầu ra: kẻ tấn công muốn tìm Thông điệp M.

Bước 1: Tính $r_1 = e_1^{-1} \ mod \ e_2$

Bước 2: Tính $r_2 = \frac{r_1.e_1 - 1}{e_2}$

Bước 3: Tính $M = C_1^{r_1}.\left(C_2^{r_2}\right)^{-1} mod\ n$

Công thức tính M ở trong bước 3 đúng bởi vì:

Có $C_1^{r_1}.\left(C_2^{r_2}\right)^{-1} mod\ n = M^{r_1.e_1}.(M^{r_2.e_2})^{-1} mod\ n$

Mặt khác:

$$C_1 = M^{e1}\ mod\ n) = M^{r_1.e_1}.\left(M^{-\frac{r_1.e_1-1}{e_2}.e_2}\right) mod\ n = M^{r_1.e_1}.(M^{1-r_1.e_1})\ mod\ n$$

$$= M^{r_1.e_1+1-r_1.e_1}\ mod\ n = M^1\ mod\ n = M\ mod\ n = M$$

Để hiểu rõ hơn sinh viên có thể xem trang 133-134/145 giáo trình mật mã học cơ sở ver2.

Nhiệm vụ 2.2

Thám mã với bản mã RSA trong file document.txt và 2 khóa công khai là public_key1.pem, public_key2.pem

Cấp quyền thực thi cho chương trình:

- chmod +x ./rsa_prime.py

Chạy ./rsa_prime.py --help để hiểu rõ cấu trúc đầu vào.

Thám mã với cú pháp sau:

- ./rsa_prime.py -n <modulus> -e1 <Exponent_1> -ct1 <Cipher_text_1> -e2 <Exponent_2> -ct2 <Cipher_text_2> [-of {decimal,hex,base64,quoted,ascii,utf-8,raw}] -q

IV.Kết quả

Vào terminal của máy khởi tạo lab chạy lệnh sau để kiểm tra kết quả tiến trình.

- checkwork

Sau khi kiểm tra đã hoàn thành bài thực hành, sinh viên chạy lệnh stoplab để kết thúc tiến trình bài thực hành.

- stoplab cryp-attak-rsa-prime