Hướng dẫn thực hành thám mã RSA mục đích 1

labtainer common_m1

I.Tổng quan

Bài tập này minh họa việc phá bản mã hóa RSA theo Phương thức Tấn công mục đích 1: phá vỡ toàn bộ hệ thống

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

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.

Phương pháp: Sử dụng khóa công khai và bí mật của mình để sinh ra khóa bí mật của người khác. Nghĩa là căn cứ vào khóa công khai của người bị tấn công, kẻ tấn công có thể tìm được khóa bí mật của người bị tấn công mà không cần biết chính xác các tham số khác $\Phi(n)$ với N là modulo chung.

Cách thức thực hiện: Để tìm được khóa bí mật này, cần phải tìm một số sao cho thỏa mãn 2 điều kiện:

- là bội của $\Phi(n)$ (ĐK1).

- Nguyên tố cùng nhau với khóa công khai (ĐK2).

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 rsacommon1.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.

Việc giải mã cần sử dụng thuật toán Eculid mở rộng - extended_gcd nên sinh viên cần bổ sung vào source code có sẵn còn thiếu.

```
def extended_gcd(a, b):#extended_Ecliud
        return gcd, r, s
def attack(c, e1, e2,d2, N):
    return m
```

Tổng quát về thuật toán eculid mở rộng.

Phương trình Đi-ô-phăng: ax+by=c

> Giải thuật Euclid mở rộng kết hợp quá trình tìm ƯCLN*(a, b)* trong thuật toán Euclid với việc tìm một cặp số x, y thoả mãn phương trình Đi-ô-phăng. Giả sử cho hai số tự nhiên a, b, ngoài ra a>b>0. Đặt *ro=a,r1=b,* chia *r0* cho *r1* được số dư *r2* và thương số nguyên *q1*. Nếu *r2=0* thì dừng, nếu *r2* khác không, chia *r1* cho *r2* được số dư *r3*,...Vì dãy các *ri* là giảm thực sự nên sau hữu hạn bước ta được số dư *rm+2=0*.

Đầu vào: kẻ tấn công có cặp khóa công khai/bí mật của kẻ tấn công (e2, d2), khóa

công khai của người bị tấn công e1.

Đầu ra: cần tìm khóa bí mật của người bị tấn công d1

Các bước tiến hành như sau:

Bước 1: Đặt t = e2.d2 – 1

Bước 2: Kẻ tấn công tiếp tục thực hiện ĐK2. Sử dụng thuật toán Euclid mở rộng

để tìm f = ƯCLN (t, e1) có cặp số (r, s) thỏa mãn công thức r*t + s*e1 = f.

Bước 3: Nếu f = 1 đặt d1’ = s và trả về kết quả d1’.

Bước 4: Nếu f khác 1 thì đặt t =t/f rồi quay lại bước 2.

Với việc thực hiện các bước trên, kẻ tấn công hoàn toàn có thể tìm được khóa bí mật

của người bị tấn công mà không mất quá nhiều thời gian và các bước tính toán.

Để hiểu rõ hơn sinh viên có thể xem trang 132/145 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 ./rsacommon1.py

Chạy ./rsacommon2.py --help để hiểu rõ cấu trúc đầu vào.

Thám mã với cú pháp sau:

- ./rsacommon1.py -n <modulus> -e1 <Exponent_1> -e2 <Exponent_2> -ct <Cipher_text> [-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 common1_m1