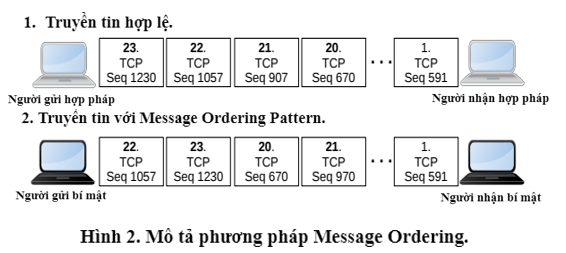
**Phát hiện giấu tin trong mạng kỹ thuật Manipulated Message Ordering**

# **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)

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*

hoặc:

wget [*https://github.com/tmanh191/steganograpy-network/raw/main/imodule.tar*](https://github.com/tmanh191/steganograpy-network/raw/main/imodule.tar)Thực hiện giải nén file imodule.tar và chuyển vào thư mục ~/labtainer/trunk/labs/

## *3.2 Thực hiện*

### *3.2.1 Task 1*

Khởi động bài lab: *labtainer -r detect\_mmo\_stegano*

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

**Cơ chế:** các số xuất hiện trong array là vị trí của các bit 1 trong message binary.  
Ví dụ binary: 11001100  
sẽ chuyển thành: 1,2,0,3,5,6,4,7

**Message -> binary -> sequence number**

**Nhiệm vụ:**

Sender gửi thành công file message.txt đã được mã hóa cho receiver

Cả 2 cần chuyển file từ tool CCEAP để bắt đầu chạy. Cả 2 máy đã có sẵn file ./script.sh và sinh viên chỉ cần chạy.

**Sender: ./script.sh**

**Receiver: ./script.sh**

Nếu cần tăng tốc độ gửi, sửa option -T 50 thành -T 30 của file ./script.sh Sender.

Yêu cần receiver phải chạy server trước.

### 3.2.2 Task 2

Ở máy **Detector** cần phải thu được các gói tin về rồi mới đi vào phân tích và giải mã. Sinh viên có thể sử dụng dump hoặc wireshark để bắt gói tin.

Hiện tại lab đã cung cấp đoạn code packet\_sniffing.py  
Khi Sender đang gửi gói tin cho receiver thì **Detector** hãy chạy lệnh:

*sudo python3 packet\_sniffing.py*

**Lưu ý:** Hãy chạy file packet\_sniffing.py trước để sẵn sàng bắt gói tin

***Detector bắt gói tin -> Receiver bật server lắng nghe -> Sender gửi gói tin***

Để tránh không bị mất gói tin nào.  
**Detector hãy lướt và xem thử các gói tin nào cần phải lọc.**

**Giải thích:** đây là đoạn code hầu như đã bắt toàn bộ gói tin trao đổi qua lại giữa receiver và sender, Sinh viên cần phải lọc gói tin.

Vấn đề của mỗi **Detector** là lọc gói tin nào bởi vì chỉ cần thiếu 1 gói thôi thì cả đoạn message sẽ bị lỗi.

Khi phân tích thì thấy gói tin đi từ sender đi tới receiver có dport =9999 và có phần load = b'n\x00\x00\x00XXXXXXXX'

Vì vậy cần phải tiến hành lọc tất cả các gói tin đi từ sender đi tơi sreceiver có dport -=9999 và có phần load

### 3.2.3 Task 3

Lab đã cung cấp đoạn code để lọc gói tin theo yêu cầu task 2

Yêu cần sinh viên sửa 3 tham số trong file packet\_sniffing \_filter.py bao gồm

*ip\_nguon = "?.?.?.?"*

*ip\_dich = "?.?.?.?"*

*port\_dich = ????*

Lưu và tiếp tục chạy lệnh:

*sudo python3 packet\_sniffing\_filter.py*

Receiver và Sender chạy lệnh ***./script.sh*** để bắt đầu truyền tin

**Lưu ý trình tự chạy:** Detector bắt gói tin -> Receiver bật server lắng nghe -> Sender gửi gói tin

Trong quá trình gửi **Detector** dễ dàng nhìn thấy phần load thay đổi liên tục.

Và nó có thể chứa thông tin. Sau khi bắt thành công toàn bộ gói tin. Dừng code và đọc file result.txt chứa toàn bộ phần giá trị của load.

### 3.2.4 Task 4

Bây giờ sinh viên cần lọc các giá trị cần thiết

b'\**x00**\x00\x00\x00XXXXXXXX'

b'\**x02**\x00\x00\x00XXXXXXXX'

b'\**x05**\x00\x00\x00XXXXXXXX'

b'\**x06**\x00\x00\x00XXXXXXXX'

b'\**x08**\x00\x00\x00XXXXXXXX'

Cần 1 đoạn code đầu vào là file result.txt và đầu ra là chỉ lấy phần in đậm.

**Giải thích:** Đây chính là giá trị mà tool của người giấu tin muốn chỉ định cho số sequence number, nghĩa là khi người gửi gửi đi tin có đoạn này thì người nhận sẽ hiểu được phần tin đậm đại diện cho số sequence number, và phần in đậm chỉ có 2 chữ số(hệ 16), chuyển nó về dạng mã ascii hệ thập phân. Vậy giới hạn của tool là 16\*16 =256. Số sequence number lớn nhất là 256 mà tool có thể gửi được.

Lab đã cung cấp đoạn code để chuyển dạng hệ 16 về dạng thập phân

**Detector chạy lệnh:**

*sudo python3 filter\_packet.py*

đọc file sequence.txt

*cat sequence.txt*

Bây giờ Detector đã có toàn bộ số sequence number mà người giấu tin đã gửi.

Sinh viên cần giải mã thành công đoạn số này

### Task 5

**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]

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 : **1,0,2,4,5,3,7,6**  
thành dãy binary gồm 8 bit: \*\*\*\*\*\*\*\*

Gợi ý: 1,2,3,0,4,5,6,7 -> 11100000 -> cắt lấy 7 bit đầu **1110000** đưa vào lệnh dưới, lấy ký tự đó làm password giải mã file decode\_mmo\_stegano.zip

Lấy 7 bit đầu tiên nhét vào lệnh dưới để tìm ra ký tự tương ứng vơi binary

Dùng lệnh: *python3 -c "print(chr(int('???????', 2)))"*

Để chuyển binary về dạng ký tự tương ứng trong bảng mã ASCII

Sinh viên sẽ thu được 1 ký tự, lấy nó để giải mã file decode\_mmo\_stegano.zip:

*unzip -P [ký tự giải mã được] decode\_mmo\_stegano.zip*

Thực hiện đọc code, đây là code mà sender và receiver dùng để mã hóa và giải mã, sau đó dùng tool để trao đổi với nhau:

*cat decode\_mmo\_stegano.py*

*python3 decode\_mmo\_stegano.py -h*

**Nhiệm vụ giải mã:**

Hãy dùng đoạn code này để giải mã file sequence.txt

## 3.3 Checkwork

Sinh viên thực hiện checkwork bằng lệnh: *checkwork*

## 3.4 Kết thúc bài

Sinh viên kết thúc bài lab: *stoplab*