HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA AN TOÀN THÔNG TIN

HỌC PHẦN: CÁC KỸ THUẬT GIẤU TIN
MÃ HỌC PHẦN: INT14102

Chủ đề: Giấu tin trong âm thanh

Lab: stego_attack_compress_paritycoding

Sinh viên thực hiện:	Lê Tuấn Anh
Mã sinh viên:	B21DCAT028
Nhóm:	04
Giảng viên hướng dẫn:	Đỗ Xuân Chợ

HÀ NỘI 2025

Bài lab Các kỹ thuật giấu tin: stego_attack_compress_paritycoding

1. Mục đích

Giúp sinh viên hiểu được thuật toán giấu tin trong âm thanh sử dụng phương pháp mã hoá chẵn lẻ (parity coding)

2. Yêu cầu đối với sinh viên

Quen thuộc với hệ điều hành Linux và có kiến thức về kỹ thuật giấu tin.

3. Nội dung lý thuyết

Parity Coding (mã hóa chẵn lẻ) là một kỹ thuật giấu tin trong âm thanh dựa trên việc điều chỉnh tính chẵn lẻ của một nhóm mẫu (sample) để mã hóa từng bit của thông điệp bí mật. Thay vì thay đổi trực tiếp giá trị của từng mẫu, kỹ thuật này nhóm các mẫu lại thành khối, tính tổng số bit 1 của nhóm và điều chỉnh bit cuối nếu cần, để đảm bảo biểu diễn chính xác thông điệp muốn giấu.

Dữ liệu âm thanh (thường là file WAV) sẽ được chia thành các nhóm mẫu. Một nhóm có thể gồm 3, 4 hoặc 5 mẫu liên tiếp, tùy theo thiết kế. Mỗi nhóm dùng để giấu 1 bit thông điệp.

Trong mỗi nhóm, ta trích bit LSB (least significant bit – bit cuối) của từng mẫu, tạo thành chuỗi bit con. Sau đó, tính tổng số bit 1 trong chuỗi đó để xác định parity:

- Nếu số bit 1 là chẵn → parity = 0
- Nếu số bit 1 là lẻ → parity = 1

Sau khi tính parity, ta so sánh với bit thông điệp cần giấu:

- Nếu parity đã trùng với bit thông điệp → không cần thay đổi
- Nếu khác nhau, ta lật (flip) LSB của một mẫu bất kỳ trong nhóm để đổi từ chẵn sang lẻ (hoặc ngược lại)

Sau khi giấu hết các bit, các nhóm được ghép lại để tạo thành file âm thanh mới chứa thông điệp.

4. Nội dung thực hành

Khởi động bài lab:

Vào terminal, gõ:

labtainer stego_attack_compress_paritycoding

(Chú ý: Sinh viên sử dụng mã sinh viên của mình để nhập thông tin email người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm)

Nhiệm vụ 1: Giấu tin vào file âm thanh sử dụng parity coding

- Sau khi khởi động xong bài lab, có 1 terminal của Alice
- Trong terminal Alice, đã có sẵn một file âm thanh input.wav, một file dữ liệu secret.txt, một tool dùng để giấu tin vào trong file âm thanh và một code python dùng để tấn công compress vào kỹ thuật giấu tin parity coding
- Sinh viên tiến hành hiển thị thông tin về cách sử dụng công cụ:

python3 parity coding_tool.py -h

- Sinh viên tiến hành sử dụng tool để giấu tin trong file message.txt vào file âm thanh input.wav. Sinh viên sử dụng câu lệnh:

python3 steg.py -e -i input.wav -m message.txt -o secret.wav

Nhiệm vụ 2: Trích xuất tin vào file âm thanh sử dụng parity coding

- Sinh viên tiến hành sử dụng tool để trích xuất tin giấu vào result.txt với file âm thanh secret.wav. Sinh viên sử dụng câu lệnh:

python3 steg.py -d -i secret.wav -o result.txt

- Xem file result.txt

cat result.txt

Nhiệm vụ 3: Tấn công compress vào file âm thanh đã được giấu tin sử dụng PARITY CODING

- Sinh viên tiến hành hiển thị thông tin về cách sử dụng công cụ tấn công compress vào parity coding:

python3 compress.py -h

- Thực hiện tấn công compress vào file âm thanh đã được giấu bằng câu lệnh:

python3 compress.py -i secret.wav -o compress.wav -r 128k

Nhiệm vụ 4: Giải mã thông tin trong file âm thanh sử dụng parity coding đã bị tấn công

- Sau khi thực hiện tấn công vào file âm thanh đã giấu tin, sinh viên thực hiện giải mã thông tin đã được giấu trong file này:

python3 steg.py -d -i compress.wav -o result2.txt

- Sinh viên sẽ bị lỗi khi thực hiện trích xuất thông tin được giấu trong file âm thanh này, chứng tỏ cuộc tấn công compress vào parity coding đã thực hiện thành công

Nhiệm vụ 5: So sánh file âm thanh gốc với file âm thanh đã bị tấn công compress vào phương pháp giấu tin bằng parity coding

- Mục đích của nhiệm vụ này là tìm hiểu sự khác biệt giữa 2 file wav trước và sau tấn công. Tại terminal của Alice, sử dụng công cụ soxi để xem thông tin kỹ thuật của 2 file âm thanh

soxi input.wav

soxi attacked_compress.wav

- Kết quả cho thấy 2 file đã có sự khác biệt thông số kỹ thuật. Tuy nhiên, điều đó không có nghĩa là nội dung âm thanh giống nhau, do đó cần so sánh raw data của 2 file âm thanh. Chúng ta thử xác định xem raw data của 2 file wav này khác nhau bắt đầu từ byte nào. Sử dụng lệnh

cmp -l <(sox input.wav -t .raw -) <(sox attacked_compress.wav -t .raw -) | head -n 20

- Kết quả cho thấy ký tự thứ 5 của dòng 1 là nơi đầu tiên xảy ra sự khác biệt. Như vậy, mặc dù 2 file có cùng định dạng và thông số kỹ thuật nhưng nội dung 2 file khác nhau hoàn toàn.

Kết thúc bài lab:

Kiểm tra checkwork:

checkwork

Trên terminal đầu tiên sử dụng câu lệnh sau để kết thúc bài lab:

stoplab