Bài thực hành: Giấu tin trong các tiêu đề HTTP

1. Khái quát chung

1.1. Tổng quan

Bài lab "stego-code-http-headers" trên Labtainer được thiết kế để giúp sinh viên thực hành kỹ thuật **giấu tin (steganography)** trong các tiêu đề HTTP. Sinh viên sẽ phân tích các tệp PCAP để trích xuất các thông điệp ẩn, sử dụng các công cụ như Wireshark hoặc Scapy.

Nhiệm vụ cuối cùng là sinh viên phải ghép các thông điệp ẩn ở từng task để tạo thành 1 FLAG hoàn chỉnh. FLAG của bài này có dạng là:

PTIT{str1_str2_str3_str4_str5} (với str1, str2, str3, str4, str5 lần lượt là các thông điệp giải mã được của các task1, task2, task3, task4, task5).

*(Ví dụ cụ thể về 1 FLAG: **PTIT{abc_xy_1234_John_HaCk}**)*

Good luck!!!!

1.2. Lý thuyết

- Trong bài lab này, giấu tin được áp dụng trong **giao thức HTTP**, cụ thể là trong các tiêu đề HTTP hoặc cấu trúc gói tin. Các kỹ thuật giấu tin trong HTTP tận dụng các trường hoặc thuộc tính của yêu cầu HTTP (như phương thức, tiêu đề, hoặc thứ tự tiêu đề) để mã hóa các bit thông điệp.
- Bài lab sử dụng 5 kỹ thuật giấu tin khác nhau, mỗi kỹ thuật được áp dụng trong một tệp PCAP:
    - **Chữ hoa/chữ thường của tiêu đề**: Sử dụng sự khác biệt giữa chữ hoa và chữ thường (ví dụ: host vs HOST) để mã hóa bit 0 hoặc 1.
    - **Thứ tự tiêu đề**: Sử dụng thứ tự xuất hiện của các tiêu đề (như Host trước User-Agent hoặc ngược lại) để mã hóa bit.
    - **Kết hợp nhiều tiêu đề**: Sử dụng nhiều tiêu đề (như Host và Accept-Encoding) để mã hóa bit, yêu cầu các tiêu đề phải nhất quán.
    - **Kết hợp phương thức và tiêu đề**: Sử dụng phương thức HTTP (GET hoặc POST) và tiêu đề (như User-Agent) để mã hóa bit.
    - **Lưu lượng nhiễu**: Nhúng thông điệp trong các gói tin hợp lệ, xen lẫn với các gói tin nhiễu để làm khó phân tích.
- Thông điệp trong bài lab được mã hóa theo quy trình sau:
    - **Chuyển văn bản thành bit**: Mỗi ký tự trong thông điệp (ví dụ: T) được chuyển thành mã ASCII (8 bit). Ví dụ: T = 84 (ASCII) = 01010100 (nhị phân).
    - **Nhúng bit vào HTTP**: Mỗi bit được mã hóa vào một thuộc tính của yêu cầu HTTP (như tiêu đề hoặc phương thức).
    - **Tạo gói tin**: Các yêu cầu HTTP được đóng gói thành các gói tin TCP/IP và lưu vào tệp PCAP.
    - **Giải mã**: Người phân tích trích xuất chuỗi bit từ các gói tin, nhóm thành các khối 8 bit, và chuyển thành ký tự ASCII.

1.3. Mục tiêu của bài lab

- Tạo các tệp PCAP từ các script Python được cung cấp.
- Phân tích các tệp PCAP để trích xuất chuỗi bit ẩn trong các tiêu đề HTTP.

- Giải mã chuỗi bit thành các thông điệp văn bản.
- Ghép các thông điệp để tạo thành thông điệp cuối cùng.
- Sử dụng Wireshark để phân tích lưu lượng mạng, nhận diện và tách thông tin giấu tin.

1.4. Yêu cầu đối với sinh viên

- Kiến thức cơ bản về giao thức HTTP, phân tích gói tin, và giấu tin.
- Có kiến thức cơ bản về hệ điều hành Linux, công cụ Wireshark.

2. Chuẩn bị môi trường thực hành

Mở terminal, trong thư mục labtainer-student, bắt đầu bài thực hành bằng lệnh:

labtainer -r stego-code-http-headers

*(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)*

Kiểm tra các bài tập phải làm trong bài thực hành

checkwork

3. Nội dung thực hành

3.1. Task 1: host_case

Kỹ thuật giấu tin: Sử dụng chữ hoa/chữ thường của tiêu đề Host trong yêu cầu HTTP:

- Nếu bit là 0, tiêu đề Host được đặt là host.
- Nếu bit là 1, tiêu đề Host được đặt là HOST.
- Các tiêu đề khác (như User-Agent, Accept) được thêm ngẫu nhiên để tăng tính thực tế.

Sinh viên thực hiện lệnh sau để tạo ra tệp PCAP: host_case.pcap

python3 host_case_pcap.py

Để phân tích file PCAP vừa mới tạo được, sinh viên chạy lệnh:

wireshark host_case.pcap &

Trên wireshark, nhập *http* vào thanh filter:

Sau đó tiến hành kiểm tra tiêu đề Host:

- Mở rộng phần Hypertext Transfer Protocol trong mỗi gói tin.
- Ghi lại:
    - host → 0.
    - HOST → 1.
- Ví dụ: Nếu thấy host, HOST, host, chuỗi bit là 010.

Sinh viên tiến hành nhập lệnh sau để giải mã chuỗi bit:

python3 decode_bits.py

3.2. Task2: header_order

Kỹ thuật giấu tin: Sử dụng thứ tự của các tiêu đề Host và User-Agent trong yêu cầu HTTP:

- Nếu bit là 0, tiêu đề Host được đặt trước User-Agent.
- Nếu bit là 1, tiêu đề User-Agent được đặt trước Host.
- Các tiêu đề khác (như Accept, X-Request-ID) được thêm để che giấu.

[redacted]

Sinh viên thực hiện lệnh sau để tạo ra tệp PCAP: header_order.pcap

python3 header_order_pcap.py

Để phân tích file PCAP vừa mới tạo được, sinh viên chạy lệnh:

wireshark header_order.pcap &

Trên wireshark, nhập *http* vào thanh filter:

Sau đó, kiểm tra thứ tự tiêu đề:

- Mở rộng phần HTTP, xem thứ tự Host và User-Agent.
- Ghi lại:
    - Host trước User-Agent → 0.
    - User-Agent trước Host → 1.

Sinh viên tiến hành nhập lệnh sau để giải mã chuỗi bit:

python3 decode_bits.py

3.3. Task3: multi_header

Kỹ thuật giấu tin: Kết hợp hai tiêu đề Host và Accept-Encoding:

- Nếu bit là 0: Host: example.com và Accept-Encoding: gzip, br.
- Nếu bit là 1: HOST: example.com và Accept-Encoding: deflate, identity.
- Cả hai tiêu đề phải nhất quán (cùng mã hóa 0 hoặc 1) trong mỗi gói tin.

Sinh viên thực hiện lệnh sau để tạo ra tệp PCAP: multi_header.pcap

python3 multi_header_pcap.py

Để phân tích file PCAP vừa mới tạo được, sinh viên chạy lệnh:

wireshark multi_header.pcap &

Trên wireshark, nhập *http* vào thanh filter:

Kiểm tra hai tiêu đề:

- Host: Host → 0, HOST → 1.

- Accept-Encoding: gzip, br → 0, deflate, identity → 1.
- Nếu cả hai cùng 0 hoặc 1, ghi lại bit tương ứng.

Sinh viên tiến hành nhập lệnh sau để giải mã chuỗi bit:

python3 decode_bits.py

3.4. Task4: combined_tech

Kỹ thuật giấu tin: Kết hợp phương thức HTTP và User-Agent:

- Nếu bit là 0: Phương thức GET, User-Agent là Windows.
- Nếu bit là 1: Phương thức POST, User-Agent là iPhone.
- IP nguồn (10.0.0.1), đích (10.0.0.2), cổng TCP 80.
- Các tiêu đề khác (như Referer, Content-Length cho POST) thêm ngẫu nhiên.

Sinh viên thực hiện lệnh sau để tạo ra tệp PCAP: combined_tech.pcap

python3 combined_tech_pcap.py

Để phân tích file PCAP vừa mới tạo được, sinh viên chạy lệnh:

wireshark combined_tech.pcap &

Trên wireshark, nhập *http* vào thanh filter:

Kiểm tra phương thức và User-Agent:

- GET và Mozilla/5.0 (Windows...) → 0.
- POST và Mozilla/5.0 (iPhone...) → 1.

Sinh viên tiến hành nhập lệnh sau để giải mã chuỗi bit:

python3 decode_bits.py

3.5. Task5: noise_traffic

Kỹ thuật giấu tin: Sử dụng phương thức HTTP với lưu lượng nhiễu:

- Nếu bit là 0: Phương thức GET, Host: example.com.
- Nếu bit là 1: Phương thức POST, Host: example.com.
- Ngẫu nhiên thêm gói tin nhiễu với phương thức GET hoặc PUT, Host: noise.com.
- IP nguồn (10.0.0.1), đích (10.0.0.2), cổng TCP 80.

Sinh viên thực hiện lệnh sau để tạo ra tệp PCAP: noise_traffic.pcap

python3 noise_traffic_pcap.py

Để phân tích file PCAP vừa mới tạo được, sinh viên chạy lệnh:

wireshark noise_traffic.pcap &

Trên wireshark, dùng bộ lọc:

http and ip.dst == 10.0.0.2 and http.host == example.com

Kiểm tra phương thức HTTP:

- GET → 0.
- POST → 1.

Sinh viên tiến hành nhập lệnh sau để giải mã chuỗi bit:

python3 decode_bits.py

3.6. Task6: check_flag

Sau khi sinh viên đã hoàn thành hết các thử thách bên trên và tìm ra được thông điệp ẩn trong các task, tiến hành chạy lệnh:

./check_flag

Sinh viên nhập FLAG tìm được theo đúng định dạng ở đầu bài đã đề cập rồi nhấn **Enter** để tiến hành kiểm tra.

4. Kết thúc bài lab

- **Kết thúc bài lab:**
    - Kiểm tra checkwork:

      checkwork

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

      stoplab

    - Khi bài lab kết thúc, một tệp zip lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab.
- **Khởi động lại bài lab:**
    - Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

      labtainer -r stego-code-http-headers