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

BÁO CÁO BÀI TẬP LỚN
HỌC PHẦN: MẬT MÃ HỌC CƠ SỞ
MÃ HỌC PHẦN: INT1344

XÂY DỰNG BÀI THỰC HÀNH
MÔ PHỎNG TẤN CÔNG BREACH
BREACH_LAB

Sinh viên thực hiện:

B22DCAT034 Trương Quốc Bảo

Giảng viên hướng dẫn: PGS.TS Đỗ Xuân Chợ

HỌC KỲ 2 NĂM HỌC 2024-2025

BÀI THỰC HÀNH: MÔ PHỎNG TẤN CÔNG BREACH

1. Giới thiệu chung về bài thực hành

Bài thực hành này tập trung vào việc mô phỏng kỹ thuật tấn công BREACH, một lỗ hổng bảo mật được công bố năm 2013. BREACH khai thác điểm yếu trong cơ chế nén HTTP (như gzip) kết hợp với các phản hồi mã hóa (HTTPS) để dần dần giải mã dữ liệu nhạy cảm (ví dụ: CSRF token, thông tin phiên đăng nhập). Kẻ tấn công có thể đoán từng phần nội dung được mã hóa thông qua phân tích kích thước phản hồi sau khi nén.

Trong bài thực hành này, chúng ta sẽ mô phỏng tấn công BREACH bằng cách khai thác tính năng nén HTTP và phân tích sự thay đổi kích thước phản hồi để thu thập thông tin nhạy cảm. Qua đó, sinh viên sẽ hiểu rõ nguy cơ tiềm ẩn khi kết hợp mã hóa (TLS/SSL) với nén dữ liệu và cách phòng chống tấn công này.

Mục đích:

- Hiểu nguyên lý hoạt động của tấn công BREACH.
- Nắm được điểm yếu khi kết hợp nén dữ liệu (HTTP compression) với mã hóa (HTTPS).
- Thực hành mô phỏng tấn công trong môi trường kiểm soát, phân tích sự thay đổi kích thước phản hồi.
- Rút ra bài học về bảo mật trong thiết kế ứng dụng web.

Yêu cầu đối với sinh viên:

- Hiểu nguyên lý hoạt động của tấn công BREACH.
- Nắm được điểm yếu khi kết hợp nén dữ liệu (HTTP compression) với mã hóa (HTTPS).
- Thực hành mô phỏng tấn công trong môi trường kiểm soát, phân tích sự thay đổi kích thước phản hồi.
- Rút ra bài học về bảo mật trong thiết kế ứng dụng web.

2. Nội dung bài thực hành

- Tải bài thực hành:

imodule https://github.com/Baodeptraii/breach_lab/raw/refs/heads/main/imodule.tar

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

- Sinh viên khởi động bài lab:

Chạy lệnh:

labtainer -r breach_lab

(Chú ý: sinh viên sử dụng <TÊN_TÀI_KHOẢN> của mình để nhập thông tin người thực hiện bài lab khi có yêu cầu, để sử dụng khi chấm điểm.)

Sau khi khởi động bài lab, hai container hiện lên bao gồm Server và Attacker sinh viên thực hiện làm theo yêu cầu.

- Terminal 1 (Server): chạy Flask server mô phỏng backend bị tấn công.
- Terminal 2 (Attacker): chạy script Python để mô phỏng hacker tấn công.

TASK 1: Kiểm tra kết nối

Trên cả 2 terminal gõ lệnh :

ifconfig

để kiểm tra địa chỉ IP.

Thực hiện kiểm tra kết nối giữa Server và Attacker bằng lệnh :

ping <địa chỉ IP>

Nếu ping giữa 2 máy thành công, chúng ta đã sẵn sàng để thực hiện mô phỏng.

TASK 2: Cài đặt môi trường và thư viện cần thiết

Kiểm tra các file có sẵn trên 2 terminal bằng lệnh :

ls

Trên terminal của Server sẽ có file Python **server.py**, file này nhằm mục đính mô phỏng lại Server.

Trên terminal của Attacker sẽ có file Python **attacker.py**, file này chứa đoạn scripts phục vụ cho việc tấn công

Trong bài này, chúng ta chỉ cần thực hiện việc chạy các file Python, đảm bài đã cài đầy đủ thư viện cần thiết bằng lệnh:

pip install flask requests

TASK 3: Mô phỏng tấn công BREACH

Trên terminal của Server, cần chỉnh sửa file **server.py** để nhận đúng địa chỉ IP khởi tạo Server, gõ lệnh:

sudo nano server.py

để thực hiện chỉnh sửa. Kéo xuống dưới và sửa *app.run(host='')* sao cho đúng với địa chỉ của máy Server.

Ta thực hiện điều tương tự với terminal bên Attacker, gõ lệnh :

sudo nano attacker.py

và sửa *mal_url=’’* thành địa chỉ IP của Server.

Sau khi đã đảm bảo các đoạn scripts được cấu hình chính xác, thực hiện khởi động Server bằng lệnh:

python3 server.py

Sinh viên quan sát *SECRET TOKEN = ‘’*. Đây là mục tiêu của cuộc tấn công. Sinh viên hoàn toàn có thể tự lựa chọn *SECRET TOKEN* của mình để thử nghiệm bằng cách chỉnh sửa trong file **server.py**.

Output sẽ cho ra địa chỉ có dạng

SECRET TOKEN = '...'

** Running on http://<ip_server>:5000.*

Sinh viên nhấn Ctrl + Chuột trái hoặc mở trình duyệt bên ngoài và truy cập đường link.

Sinh viên thay đổi đường link thành:

http://<địa chỉ Server>:5000/poc/?request_token=1

và quan sát trên màn hình.

Trên terminal của Attacker thực hiện lệnh :

python3 attacker.py

để thực hiện tấn công BREACH, hãy quan sát output. Script tự động dự đoán từng byte của *SECRET TOKEN* dựa trên chênh lệch Content-Length. Khi tấn công thành công, thông báo *FOUND BYTE* sẽ xuất hiện cùng các byte đã tìm được thành công.

Đồng thời, quan sát lưu lượng gửi các yêu cầu **GET** và **POST** trên terminal của máy Server. Kiểm tra Attacker đã gửi những yêu cầu như thế nào.

Giải thích logic tấn công

- BREACH khai thác nén GZIP: khi chuỗi giử~u token và user input giống nhau, dữ liệu sẽ được nén tốt hơn → Content-Length ngắn hơn.
- Script attacker gửi 16 giá trị hex (0 → f) để xem phản hồi nào là ngắn nhất.

Sinh viên thực hiện mô phỏng và trả lời các câu hỏi sau :

- BREACH dùng kỹ thuật gì để đoán được các byte ?
- Tạo sao không đoán được byte đầu tiên của TOKEN ?
- Số “1” trong *request_token* có ý nghĩa gì ?

TASK 4: Tự tùy chỉnh token

Để hiểu kĩ hơn cách hoạt động của BREACH, sinh viên có thể tùy chỉnh lại cấu hình của các file Python để thử nghiệm.

- File server.py với dòng: *SECRET_TOKEN = '123456789abcdef'*
- File attacker.py với dòng: *HEXVAL = list('1234567890abcdef')*
- Hoặc thử thay đổi *request_token* trong đường dẫn trên Firefox thành một giá trị khác.

- Kết thúc lab:

Trên terminal khởi động lab, sinh viên sử dụng lệnh:

stoplab

Khi bài lab kết thúc, một tệp lưu kết quả được tạo và lưu vào một vị trí được hiển thị bên dưới stoplab. Sinh viên cần nộp file .lab để chấm điểm.

- Để kiểm tra kết quả khi trong khi làm bài thực hành sử dụng lệnh:

checkwork <tên bài thực hành>

Sinh viên cần nộp file .lab để chấm điểm.

Kiểm tra kết quả trong quá trình làm bài:

checkwork <tên bài lab>

- Khởi động lại bài lab: Trong quá trình làm bài sinh viên cần thực hiện lại bài lab, dùng câu lệnh:

labtainer -r breach_lab