BỘ GIAO THÔNG VẬN TẢI

**TRƯỜNG ĐẠI HỌC GIAO THÔNG VẬN TẢI TP. HỒ CHÍ MINH**

**Viện Công nghệ thông tin và Điện, Điện tử**

-----□□🕮□□-----

**BÀI TẬP LỚN MÔN AN TOÀN THÔNG TIN**

**NGHIÊN CỨU CÔNG CỤ NETCAT TRÊN KALI LINUX THU THẬP THÔNG TIN**

**GVHD: Ths. Nguyễn Văn Đôn**

**LỚP: 010412303303 NHÓM: 25**

**SV:**

|  |  |  |
| --- | --- | --- |
| **1** | 079305001883 | Huỳnh Thị Ngọc Loan |
| **2** | 082305007043 | Trần Thục Trinh |
| **3** | 054206004940 | Nguyễn Văn Sơn |
| **4** | 075206023506 | Trần Hà Đức Huỳnh |

*Tp. Hồ Chí Minh, Tháng 4 năm 2025*

**Nhận xét và đánh giá của giảng viên**

Nhận xét:

Điểm đánh giá:

Chữ ký của giảng viên

(Ký tên)

**Nguyễn Văn Đôn**

**LỜI CẢM ƠN**

Trong suốt quá trình thực hiện bài báo cáo với đề tài **“Nghiên cứu công cụ Netcat trên Kali Linux thu thập thông tin”**, em xin chân thành gửi lời cảm ơn đến thầy đã tận tình giảng dạy, định hướng và hỗ trợ em trong quá trình tìm hiểu và hoàn thiện nội dung báo cáo.

Những kiến thức mà thầy truyền đạt, cùng với những góp ý trong quá trình học tập, đã giúp em hiểu sâu hơn về các công cụ an ninh mạng, đặc biệt là cách ứng dụng Netcat trong hoạt động thu thập thông tin. Đây là những kinh nghiệm quý báu giúp em nâng cao tư duy phân tích, khả năng thực hành và tiếp cận thực tế trong lĩnh vực an toàn thông tin.

**MỤC LỤC**

[CHƯƠNG 1: TỔNG QUAN 1](#_Toc214875595)

[1.1 Bối cảnh an ninh mạng trong nước và quốc tế 1](#_Toc214875596)

[1. 1. 1. Tình hình quốc tế: 1](#_Toc214875597)

[1. 1. 2. Vấn đề cần giải quyết: 1](#_Toc214875598)

[1.2 Giới thiệu công cụ Netcat 3](#_Toc214875599)

[1. 2. 1. Netcat là gì — nguyên lý hoạt động cơ bản 3](#_Toc214875600)

[1. 2. 2. Các biến thể: netcat-traditional, netcat-openbsd, ncat (Nmap) 5](#_Toc214875601)

[1. 2. 3. Ứng dụng chính trong thu thập thông tin 8](#_Toc214875602)

[1.3 Mục tiêu nghiên cứu và phạm vi 8](#_Toc214875603)

[1. 3. 1. Mục tiêu nghiên cứu 8](#_Toc214875604)

[1. 3. 2. Phạm vi nghiên cứu 8](#_Toc214875605)

[CHƯƠNG 2: CƠ SỞ LÝ THUYẾT 10](#_Toc214875606)

[2.1 Tổng quan về nguyên tắc mạng liên quan 10](#_Toc214875607)

[2.1.1 Mở đầu 10](#_Toc214875608)

[2.1.2 ARP (Address Resolution Protocol) 10](#_Toc214875609)

[2.1.3 IP (Internet Protocol) 10](#_Toc214875610)

[2.1.4 TCP (Transmission Control Protocol) 11](#_Toc214875611)

[2.1.5 UDP (User Datagram Protocol) 11](#_Toc214875612)

[2.1.6 Sơ đồ tổng quát (tương tác giữa các layer liên quan) 11](#_Toc214875613)

[2.1.7 Ý nghĩa thực hành cho thu thập thông tin 12](#_Toc214875614)

[2.2 Kiến thức liên quan tới thu thập thông tin 12](#_Toc214875615)

[2.2.1 Khái niệm Reconnaissance 12](#_Toc214875616)

[2.2.2 Banner grabbing — mục tiêu và phương pháp 12](#_Toc214875617)

[2.2.3 Port scanning 13](#_Toc214875618)

[2.2.4 Công cụ điển hình và ví dụ lệnh 13](#_Toc214875619)

[2.2.5 Quy trình recon gợi ý để thực nghiệm 14](#_Toc214875620)

[2.2.6 Hạn chế, rủi ro và lưu ý an toàn 14](#_Toc214875621)

[2.3 So sánh Netcat với các công cụ tương tự 14](#_Toc214875622)

[2.3.1. Netcat (nc) so với Ncat (ncat) 16](#_Toc214875623)

[2.3.2. Netcat/Ncat so với Socat 17](#_Toc214875624)

[2.3.3. Netcat/Ncat so với hping3 17](#_Toc214875625)

[2.4 Vấn đề an toàn, pháp lý và đạo đức khi dùng Netcat 18](#_Toc214875626)

[2.4.1 Vấn đề an toàn: Phát hiện, Dấu vết và Mã hóa 18](#_Toc214875627)

[2.4.2 Vấn đề Pháp lý và Quy định 19](#_Toc214875628)

[2.4.3 Vấn đề Đạo đức 20](#_Toc214875629)

[CHƯƠNG 3: TRIỂN KHAI CÔNG CỤ NETCAT 22](#_Toc214875630)

[3.1 Khảo sát host và banner grabbing 22](#_Toc214875631)

[3.1.1 Mục tiêu 22](#_Toc214875632)

[3.1.2 Công cụ sử dụng 22](#_Toc214875633)

[3.1.3 Các bước thực hiện 22](#_Toc214875634)

[3.1.4 Kết quả 24](#_Toc214875635)

[3.2 Kiểm tra trạng thái cổng (Port check / Connect scan) 24](#_Toc214875636)

[3.2.1 Mục tiêu 24](#_Toc214875637)

[3.2.2 Các bước thực hiện 25](#_Toc214875638)

[3.2.3 Kết quả 26](#_Toc214875639)

[3.3 Lắng nghe và ghi nhận kết nối (Listen / Logging) 27](#_Toc214875640)

[3.3.1 Mục tiêu 27](#_Toc214875641)

[3.3.2 Các bước thực hiện 27](#_Toc214875642)

[3.3.3 Kết quả 28](#_Toc214875643)

[3.4 Truyền file đơn giản qua Netcat (Sender/Receiver) 28](#_Toc214875644)

[3.4.1 Mục tiêu 29](#_Toc214875645)

[3.4.2 Các bước thực hiện 29](#_Toc214875646)

[3.4.3 Kết quả 30](#_Toc214875647)

[3.5 Kết hợp Netcat với công cụ khác (pipeline / xử lý) 30](#_Toc214875648)

[3.5.1 Mục tiêu 31](#_Toc214875649)

[3.5.2 Các bước thực hiện 31](#_Toc214875650)

[3.5.3 Kết quả 32](#_Toc214875651)

[CHƯƠNG 4: TỔNG KẾT VÀ ĐÁNH GIÁ 34](#_Toc214875652)

[4.1 Nhận xét về quá trình thực hiện 34](#_Toc214875653)

[4. 1. 1. Thuận lợi 34](#_Toc214875654)

[4. 1. 2. Khó khăn 34](#_Toc214875655)

[4. 1. 3. Đánh giá kết quả thực nghiệm 35](#_Toc214875656)

[4.2 Kết luận chung 36](#_Toc214875657)

[Tài liệu tham khảo 37](#_Toc214875658)

**DANH MỤC HÌNH ẢNH**

[Hình 1. Khởi tạo test sever 22](#_Toc214875673)

[Hình 2. Banner grabbing 23](#_Toc214875674)

[Hình 3. Kết quả thu thập banner 24](#_Toc214875675)

[Hình 4. Kiểm tra dịa chỉ IP của máy Kali Linux 25](#_Toc214875676)

[Hình 5. Kiểm tra địa chỉ IP của máy Window 25](#_Toc214875677)

[Hình 6. Kiểm tra các port lắng 26](#_Toc214875678)

[Hình 7. Kiểm tra port ngẫu nhiên 26](#_Toc214875679)

[Hình 8. Tạo tập tin shell script 27](#_Toc214875680)

[Hình 9. Kết quả lắng và ghi nhận kết nối ở chế độ listener 28](#_Toc214875681)

[Hình 10. Chuẩn bị tệp tin cho máy gửi 29](#_Toc214875682)

[Hình 11. Thiết lập listener trên máy nhận 29](#_Toc214875683)

[Hình 12. Gửi tệp tin từ máy gửi 29](#_Toc214875684)

[Hình 13. Kết quả truyền tải 30](#_Toc214875685)

[Hình 14. Kết quả thực hiện 32](#_Toc214875686)

**DANH MỤC BẢNG**

[Bảng 1 So sánh nhanh (tính năng — ưu/nhược) 5](#_Toc214571383)

[Bảng 2 Sơ đồ tổng quát tương tác giữa các layer 11](#_Toc214571384)

[Bảng 3 Bảng so sánh Netcat với các tiện ích phổ biến khác như Ncat, Socat và hping3 15](#_Toc214571385)

[Bảng 4 Chức năng và thành phần lọc dữ liệu 31](#_Toc214571386)

# CHƯƠNG 1: TỔNG QUAN

## Bối cảnh an ninh mạng trong nước và quốc tế

An ninh mạng đang trở thành một vấn đề cấp thiết trong bối cảnh công nghệ thông tin phát triển mạnh mẽ. Các giải pháp bảo mật truyền thống không còn đủ hiệu quả trước các mối đe dọa ngày càng tinh vi.

### Tình hình quốc tế:

* **Quy mô và tốc độ tăng trưởng**: Các cuộc tấn công mạng toàn cầu đã tăng 30% trong quý 2 năm 2024, với trung bình 1.626 cuộc tấn công hàng tuần đối với mỗi tổ chức, đánh dấu mức tăng 75% so với cùng kì năm 2023. Khoảng 2,200 cuộc tấn công mạng xảy ra hàng ngày trên toàn thế giới, tương đương với mỗi một cuộc tấn công cứ mỗi 39 giây.
* **Chi phí thiệt hại khổng lồ**: Chi phí trung bình toàn cầu của một vụ vi phạm dữ liệu trong năm 2024 là 4,88 triệu USD, tăng 10% so với năm trước đó.Các nhóm nghiên cứu ước tính chi phí tội phạm mạng có thể đạt 10,5 nghìn tỷ USD vào năm 2025, trong khi một dự báo khác đề cập đến con số 23 nghìn tỷ USD vào năm 2027. Chi phí trung bình của một vụ vi phạm dữ liệu tại Hoa Kỳ là 10,22 triệu USD vào năm 2025 - mức cao nhất từ trước đến nay cho bất kỳ khu vực nào.
* **Thời gian phát hiện và xử lý**: Các nhóm bảo mật mất trung bình 277 ngày để xác định và ngăn chặn một vụ vi phạm dữ liệu, trong khi các vụ vi phạm liên quan đến thông tin đăng nhập bị mất hoặc bị đánh cắp mất 328 ngày để xác định và ngăn chặn.

### Vấn đề cần giải quyết:

* **Vấn đề phát hiện sớm trong Giai đoạn Reconnaissance:** Thời gian phát hiện vi phạm trung bình là 277 ngày, trong khi các cuộc tấn công xảy ra mỗi 39 giây. Khoảng cách thời gian này cho thấy các tổ chức chỉ phát hiện được tấn công khi đã bị khai thác sâu (giai đoạn exploitation hoặc exfiltration), chứ không phát hiện được ở giai đoạn đầu tiên - reconnaissance (thu thập thông tin). Theo mô hình Cyber Kill Chain, reconnaissance là giai đoạn bắt buộc mà kẻ tấn công phải thực hiện để:   
  Xác định mục tiêu (target identification), quét lỗ hổng bảo mật (vulnerability scanning), liệt kê dịch vụ đang chạy (service enumeration), thu thập thông tin phiên bản (version information), lập bản đồ cấu trúc mạng (network mapping), nếu phát hiện được reconnaissance, có thể ngăn chặn 100% tấn công trước khi chúng xảy ra. Tuy nhiên, hiện tại các tổ chức: không có phương pháp hiệu quả để detect reconnaissance activities. Thiếu hiểu biết về công cụ và kỹ thuật mà kẻ tấn công sử dụng Không phân biệt được legitimate traffic vs malicious reconnaissance. Tập trung quá nhiều vào phòng thủ các giai đoạn sau
* Vấn đề khoảng cách kiến thức Offensive và Defensive Security: Một trong những thách thức lớn trong lĩnh vực an ninh mạng hiện nay là sự chênh lệch kiến thức và tư duy giữa hai nhóm: Offensive Security (đội tấn công) và Defensive Security (đội phòng thủ).

**Offensive Security** (Red Team, Penetration Testers):

* Có hiểu biết sâu về cách hoạt động của các công cụ tấn công như Netcat, Nmap, Metasploit, Burp Suite…
* Nắm rõ kỹ thuật khai thác lỗ hổng (exploitation), leo thang đặc quyền (privilege escalation), di chuyển ngang (lateral movement)…
* Thường xuyên cập nhật các kỹ thuật tấn công mới.
* Tư duy chính: *“Làm thế nào để vượt qua các biện pháp phòng thủ?”*

**Defensive Security** (Blue Team, SOC Analysts):

* Chủ yếu có xu hướng phản ứng sau khi sự cố xảy ra dựa trên signatures, rules hoặc cảnh báo do các hệ thống cung cấp.
* Thiếu hiểu biết sâu về cơ chế hoạt động của công cụ tấn công nên gặp khó khăn khi truy tìm dấu vết hoặc phân tích log, lưu lượng mạng sau sự cố.
* Tư duy chính: *“Làm thế nào để phát hiện mối đe dọa đã biết?”*
* Dễ bị động trước các phương thức tấn công mới mà công cụ phòng thủ chưa cập nhật.
* Hệ quả của sự bất đối xứng này là đội phòng thủ có thể bỏ sót các dấu hiệu tấn công tinh vi, trong khi đội tấn công luôn chủ động tìm cách vượt qua các cơ chế bảo vệ.
* **Vấn đề thiếu Framework đánh giá công cụ Reconnaissance:** Hiện nay, lĩnh vực reconnaissance chưa có một framework chuẩn để đánh giá hiệu quả các công cụ. Không có tiêu chí rõ ràng để xác định công cụ nào hiệu quả nhất, công cụ nào ít bị phát hiện nhất hay công cụ nào thu thập thông tin chính xác hơn. Bên cạnh đó, cũng thiếu phương pháp luận thống nhất để so sánh công bằng giữa các công cụ, bao gồm tiêu chí đo lường (thời gian, độ chính xác, mức độ bị phát hiện), thiết lập môi trường thử nghiệm và khả năng áp dụng kết quả cho nhiều hệ thống khác nhau. Ngoài ra, chưa có taxonomy chính thức để phân loại các kỹ thuật reconnaissance theo mức độ chủ động (passive/active), mức độ stealth và loại thông tin thu thập. Điều này khiến việc lựa chọn và đánh giá các công cụ, bao gồm Netcat, trở nên khó khăn và thiếu nhất quán.
* **Tại sao lại sử dụng Netcat**: Netcat được sử dụng rộng rãi trong các cuộc tấn công mạng do khả năng quét cổng, thu thập thông tin dịch vụ và thiết lập kết nối từ xa một cách nhanh chóng và linh hoạt. Trong bối cảnh số lượng tấn công mạng gia tăng mạnh và thời gian phát hiện vi phạm kéo dài, việc hiểu rõ cách kẻ tấn công sử dụng những công cụ cơ bản như Netcat trở nên cần thiết. Nghiên cứu Netcat giúp người làm an ninh mạng nhận diện hành vi reconnaissance, phát hiện sớm dấu hiệu xâm nhập và tăng cường khả năng phòng thủ. Đây là bước quan trọng để giảm thiểu rủi ro và bảo vệ hệ thống thông tin hiệu quả hơn.

## Giới thiệu công cụ Netcat

### Netcat là gì — nguyên lý hoạt động cơ bản

* **Khái niệm**:

Netcat (nc) là một tiện ích mạng hoạt động trên dòng lệnh, cho phép đọc và ghi dữ liệu qua các kết nối TCP hoặc UDP. Công cụ này hoạt động ở tầng Transport (Layer 4) của mô hình OSI và có khả năng thiết lập kết nối theo mô hình client–server. Netcat có thể lắng nghe trên một cổng để chờ kết nối vào (bind) hoặc chủ động kết nối đến máy khác (reverse).

Khi sử dụng TCP, Netcat tạo kết nối tin cậy thông qua cơ chế bắt tay ba bước và kiểm soát luồng, phù hợp cho việc truyền dữ liệu ổn định hoặc duy trì phiên làm việc. Với UDP, Netcat hoạt động theo kiểu không kết nối (connectionless), nhanh nhưng kém tin cậy hơn, thích hợp cho các tác vụ kiểm tra và gửi dữ liệu đơn giản.

Nhờ khả năng tương tác trực tiếp với socket ở tầng Transport mà không cần lập trình phức tạp, Netcat hỗ trợ nhiều chức năng: kiểm tra kết nối, quét cổng, thu thập banner dịch vụ, truyền file, tạo đường hầm (tunneling), proxy tạm thời và thậm chí thiết lập backdoor trong môi trường kiểm thử.

* **Nguyên lí hoạt động**:

Netcat hoạt động dựa trên cơ chế ***socket programming***, tạo socket để thiết lập kênh truyền thông hai chiều giữa hai điểm cuối trong mạng. Ở chế độ ***listener*** (sử dụng tham số -l), Netcat tạo một socket, gán (bind) nó với địa chỉ IP và cổng xác định, sau đó chuyển sang trạng thái lắng nghe để chờ kết nối đến. Khi có client kết nối, Netcat thực hiện thao tác ***accept*** và thiết lập phiên giao tiếp.

Ở chế độ ***client***, Netcat tạo socket và chủ động gửi yêu cầu kết nối đến IP và cổng đích. Sau khi kết nối được thiết lập, Netcat hoạt động như một ***pipeline đơn giản***: đọc dữ liệu từ ***STDIN*** để gửi qua socket và đồng thời nhận dữ liệu từ socket để ghi ra ***STDOUT***. Quá trình truyền nhận diễn ra liên tục cho đến khi một trong hai phía đóng kết nối.

Với **TCP**, Netcat duy trì phiên làm việc ổn định, đảm bảo thứ tự và độ tin cậy của gói tin thông qua cơ chế ACK và sequence number. Với **UDP**, Netcat gửi và nhận các datagram độc lập mà không kiểm tra trạng thái hay độ tin cậy, phù hợp cho các tác vụ truyền nhanh và không yêu cầu xác nhận.

Nhờ cơ chế hoạt động đơn giản nhưng linh hoạt, Netcat có thể thực hiện nhiều chức năng như kiểm tra dịch vụ, truyền dữ liệu, thu thập thông tin, hoặc thiết lập kênh kết nối phục vụ kiểm thử bảo mật.

### Các biến thể: netcat-traditional, netcat-openbsd, ncat (Nmap)

Netcat (thường gọi nc) có nhiều triển khai (implementation) khác nhau, mỗi biến thể được phát triển với mục tiêu và ưu tiên khác nhau. Ba biến thể phổ biến nhất hiện nay là **netcat-traditional**, **netcat-openbsd**, và **ncat (thuộc bộ Nmap)**. Dưới đây tóm tắt sự khác nhau chính về chức năng, bảo mật và ứng dụng thực tế.

**Mô tả ngắn**

* **netcat-traditional**: Triển khai ban đầu, thiết kế đơn giản, mạnh mẽ cho scripting và demo. Hỗ trợ một số tính năng “tiện lợi” nhưng cũng chứa những lựa chọn có nguy cơ bảo mật (ví dụ khả năng chạy shell/command từ socket).
* **netcat-openbsd**: Phiên bản hiện đại hơn theo triết lý bảo mật của OpenBSD — nhiều tùy chọn nguy hiểm bị loại bỏ hoặc hạn chế, codebase được chú trọng bảo trì và an toàn hơn. Thường là bản nc mặc định trên nhiều bản Linux/BSD hiện đại.
* **ncat (Nmap)**: Công cụ do dự án Nmap phát triển; hướng tới tính năng phong phú, tích hợp (ví dụ SSL/TLS, proxy, authentication), phù hợp cho môi trường kiểm thử hoặc cần tính năng bảo mật/đa năng hơn netcat gốc.

Bảng 1 So sánh nhanh (tính năng — ưu/nhược)

|  |  |  |  |
| --- | --- | --- | --- |
| **Tiêu chí** | **netcat-traditionl** | **netcat-openbsd** | **ncat (Nmap)** |
| Triết lý thiết kế | Đơn giản, “toy-like”, script-friendly | An toàn, tối giản các chức năng rủi ro | Đa tính năng, tích hợp nâng cao |
| Khả năng chạy lệnh từ socket (remote exec) | Thường hỗ trợ (ví dụ option -e) tiện nhưng nguy cơ | Không khuyến khích / thường bỏ hoặc hạn chế vì lý do bảo mật | Hỗ trợ các cách thực thi an toàn hơn (ví dụ --sh-exec, --exec) với kiểm soát |
| Hỗ trợ SSL/TLS | Không (mặc định) | Không (mặc định) | Có (chứng thực TLS/SSL, --ssl) |
| Proxy / chuyển tiếp / relay | Hạn chế | Hạn chế | Hỗ trợ proxy, relay, tunneling nâng cao |
| Tính tương thích flags / cú pháp | Rất nhiều biến thể giữa bản distro | Cú pháp khá nhất quán trên hệ BSD/Linux mới | Cú pháp giống Nmap, có nhiều option mở rộng |
| Mục đích sử dụng phù hợp | Script nhanh, PoC, truyền file đơn giản | Kiểm thử an toàn, dùng ở môi trường sản xuất nơi tránh rủi ro | Khi cần mã hóa, xác thực, proxy, tích hợp sâu trong công cụ kiểm thử |
| Bảo mật | Ít hạn chế — có thể bị lạm dụng | Ưu tiên an toàn, loại bỏ lỗ hổng lộ shell | Tích hợp cơ chế an toàn (TLS,auth), thích hợp cho môi trường nghiêm túc |

* **Ghi chú về flags / hành vi**Mỗi bản nc có tập flags hơi khác nhau tùy bản phân phối (Debian/Ubuntu, CentOS, OpenBSD, v.v.). Một số điểm phổ biến cần lưu ý trong báo cáo:
* Một số bản *traditional* cung cấp tùy chọn chạy chương trình khi có kết nối (ví dụ tham số kiểu -e), tính năng này rất tiện cho demo nhưng là rủi ro bảo mật nếu dùng trên mạng không được phép. Vì vậy nhiều bản hiện đại (ví dụ netcat-openbsd) đã loại hoặc vô hiệu hoá nó.
* ncat mở rộng bằng các option hỗ trợ **SSL/TLS**, **proxy**, **xử lý đa kết nối**, và các tùy chọn exec an toàn hơn — phù hợp khi cần kênh mã hoá hoặc tích hợp với hệ thống kiểm thử phức tạp.
* Khi viết script hoặc đưa vào báo cáo, **luôn** ghi rõ phiên bản nc (vd: nc (OpenBSD) 1.XXX hoặc ncat 7.X) và khuyến nghị đọc man nc/man ncat để biết cú pháp chính xác.
* **Khi nào chọn bản nào (đề xuất cho thực nghiệm trong báo cáo)**
* Nếu mục tiêu là minh họa nguyên lý, viết script quét/transfer đơn giản trong lab: **netcat-traditional** hoặc **netcat-openbsd** đều ổn. Ưu tiên **netcat-openbsd** nếu muốn tránh những tính năng rủi ro.
* Nếu cần **mã hóa** kênh, proxy, hoặc các test phức tạp tương tác với dịch vụ có xác thực: dùng **ncat** (vì hỗ trợ TLS, gzip, proxy, và nhiều tuỳ chọn quản lý kết nối).
* Trong báo cáo, nên đưa ra cả ba ví dụ: một ví dụ banner grabbing/transfer với netcat-openbsd, và một ví dụ tương đương với ncat có --ssl để so sánh kết quả và chỉ rõ ưu/nhược.
* **Ví dụ minh họa (ghi ý — kiểm tra man page trước khi chạy)**
* Banner grabbing (ví dụ, concept — cú pháp có thể khác giữa bản):
  + echo -e "HEAD / HTTP/1.0\r\n\r\n" | nc target 80
* Truyền file (listener trên máy nhận):
  + nc -l -p 1234 > file.out (cú pháp có thể khác: -l -p hoặc -l kết hợp -p tuỳ bản)
* Với ncat (mã hoá SSL):
  + ncat --ssl --listen -p 4443 --sh-exec "cat > file.out"

**Lưu ý:** các ví dụ trên mang tính minh họa — trước khi dùng trên máy bạn, kiểm tra man nc hoặc ncat --help vì tham số có thể khác giữa các bản cài đặt.

### Ứng dụng chính trong thu thập thông tin

* **Banner grabbing**: kết nối tới cổng dịch vụ để đọc thông tin chào (banner) mà service trả về (phiên bản, server type).
* **Kiểm tra cổng đơn/nhỏ (simple port checking)**: kiểm tra xem cổng có mở hay không (nhanh, lightweight).
* **Probing dịch vụ bằng tay**: gửi thủ công các yêu cầu (HTTP, SMTP, FTP…) để xem phản hồi chi tiết.
* **Chuyển/nhận dữ liệu đơn giản**: tải về/đẩy file trong môi trường kiểm thử (lab).
* **Tạo listener để chẩn đoán**: lắng nghe một cổng để kiểm tra kết nối từ client/tests.
* **Proxying / tunneling đơn giản**: nối hai socket (giới hạn, thường dùng cho debug).

## Mục tiêu nghiên cứu và phạm vi

### Mục tiêu nghiên cứu

Mục tiêu của đề tài là nghiên cứu công cụ Netcat trên hệ điều hành Kali Linux và tìm hiểu cách sử dụng công cụ này trong hoạt động thu thập thông tin mạng một cách hợp pháp. Thông qua quá trình nghiên cứu và thực nghiệm, đề tài hướng tới việc:

* Làm rõ nguyên lý hoạt động và các chức năng cơ bản của Netcat.
* Ứng dụng Netcat để thực hiện các thao tác thu thập thông tin như: kiểm tra trạng thái cổng, thu thập banner dịch vụ, kết nối và ghi nhận dữ liệu trên mạng.
* Ghi nhận, phân tích kết quả thực nghiệm và rút ra những nhận xét về tính hiệu quả cũng như những hạn chế khi sử dụng Netcat trong giai đoạn thu thập thông tin.

### Phạm vi nghiên cứu

Phạm vi của đề tài tập trung vào việc nghiên cứu và thực nghiệm công cụ Netcat trong giới hạn chức năng phục vụ giai đoạn thu thập thông tin mạng. Cụ thể, đề tài chỉ xem xét:

* Các thao tác sử dụng Netcat để thiết lập kết nối đến cổng dịch vụ và quan sát phản hồi.
* Việc áp dụng Netcat cho mục đích banner grabbing, kiểm tra trạng thái cổng và ghi nhận dữ liệu truyền nhận trong môi trường kiểm thử.
* Các phân tích liên quan đến hiệu quả, giới hạn và rủi ro khi sử dụng Netcat trong thu thập thông tin, không mở rộng sang các kỹ thuật khai thác hay tấn công mạng ở các giai đoạn tiếp theo.

# CHƯƠNG 2: CƠ SỞ LÝ THUYẾT

## 2.1 Tổng quan về nguyên tắc mạng liên quan

### 2.1.1 Mở đầu

Trong các hoạt động phân tích mạng và thu thập thông tin, việc nắm vững nguyên tắc hoạt động của các giao thức cơ bản và vị trí của chúng trong mô hình OSI là tiền đề bắt buộc. Phần này trình bày các khái niệm cốt lõi — **ARP, IP, TCP, UDP** — và giải thích tầm quan trọng của chúng khi thực hiện các kỹ thuật reconnaissance, port scanning và banner grabbing.

### 2.1.2 ARP (Address Resolution Protocol)

* **Chức năng:** ánh xạ địa chỉ IP (Layer 3) sang địa chỉ MAC (Layer 2) trong mạng LAN, cho phép node gửi frame tới đúng thiết bị vật lý.
* **Cơ chế hoạt động:** khi cần gửi tới IP đích trong cùng LAN mà không có entry trong ARP cache, host phát ARP Request (broadcast); host có IP đích trả ARP Reply với địa chỉ MAC.
* **Tổn tại rủi ro:** ARP không có chứng thực — dễ bị khai thác bằng **ARP spoofing/poisoning**, khiến traffic bị chuyển hướng hoặc bị nghe lén.
* **Vị trí OSI:** hoạt động ở **Tầng liên kết dữ liệu (Layer 2)** nhưng tương tác trực tiếp với thông tin ở Layer 3 (IP).

### 2.1.3 IP (Internet Protocol)

* **Chức năng:** định tuyến, phân mảnh gói, phân biệt mạng con; địa chỉ IP dùng để xác định endpoint logic trên mạng.
* **Vị trí OSI:** **Layer 3 (Network)**.
* **Ý nghĩa với thí nghiệm:** quyết định cách thức host discovery khi vượt ra ngoài LAN (ICMP/ping, traceroute).

### 2.1.4 TCP (Transmission Control Protocol)

* **Đặc điểm chính:** giao thức có kết nối, đảm bảo độ tin cậy, thứ tự và không mất mát dữ liệu.
* **Thiết lập kết nối:** 3-way handshake (SYN → SYN/ACK → ACK).
* **Ảnh hưởng với scanning:** trạng thái cổng TCP (open/closed/filtered) được xác định dựa trên phản hồi của handshake; các kỹ thuật quét (connect, SYN/half-open, FIN, NULL, Xmas) khai thác cách TCP stack trả lời để suy luận trạng thái port.
* Vị trí OSI: Layer 4 (Transport).

### 2.1.5 UDP (User Datagram Protocol)

* **Đặc điểm chính:** không kết nối, nhẹ, không đảm bảo thứ tự hay tái truyền; không có handshake.
* **Khó khăn khi quét:** nhiều dịch vụ UDP không phản hồi khi open, nên UDP scan thường trả kết quả open|filtered; cần thời gian và kết hợp kiểm tra bổ sung.
* Vị trí OSI: Layer 4 (Transport).

### 2.1.6 Sơ đồ tổng quát (tương tác giữa các layer liên quan)

Bảng 2 Sơ đồ tổng quát tương tác giữa các layer

|  |  |  |
| --- | --- | --- |
| **Thành phần** | **Vai trò** | **OSI layer** |
| ARP | Ánh xạ IP → MAC trong LAN | Layer 2 |
| IP | Định tuyến, địa chỉ mạng | Layer 3 |
| TCP | Giao tiếp có kết nối, port, handshake | Layer 4 |
| UDP | Giao tiếp không kết nối | Layer 4 |
| HTTP/SMTP/FTP... | Giao thức ứng dụng chứa banner/payload | Layer 7 |

### 2.1.7 Ý nghĩa thực hành cho thu thập thông tin

* **Host discovery (LAN):** ARP scan (arping, arp-scan) thường nhanh và chính xác để xác định host alive.
* **Chọn phương pháp quét phù hợp:** hiểu TCP/UDP giúp quyết định dùng TCP connect, SYN hay UDP scan; giảm false-positive/negative.
* **Phân tích gói:** khi cần xác minh banner hoặc phân tích payload, bắt gói ở tầng liên kết (tcpdump/Wireshark) cung cấp ngữ cảnh (ARP, IP, ports, payload)

## 2.2 Kiến thức liên quan tới thu thập thông tin

### 2.2.1 Khái niệm Reconnaissance

* **Định nghĩa:** reconnaissance (recon) là giai đoạn thu thập thông tin về mục tiêu nhằm hiểu cấu trúc mạng, host tồn tại, dịch vụ đang chạy, phiên bản phần mềm, và các điểm yếu tiềm năng.
* **Hai dạng:**
  + **Passive recon:** thu thập từ nguồn công khai hoặc nghe lén traffic trong mạng bạn kiểm soát; ưu thế là ít bị phát hiện.
  + **Active recon:** tương tác trực tiếp với mục tiêu (ping, scan, banner grabbing); thu được dữ liệu chính xác hơn nhưng dễ để lại dấu vết.

**Ghi chú đạo đức:** tất cả hoạt động active recon phải được phép; thực nghiệm chỉ tiến hành trên môi trường lab hoặc hệ thống sở hữu/được cho phép.

### 2.2.2 Banner grabbing — mục tiêu và phương pháp

* **Mục tiêu:** thu thập “banner” — thông tin mô tả service (tên phần mềm, phiên bản, đôi khi hệ điều hành) để phục vụ quá trình fingerprinting và tra cứu CVE liên quan.
* **Phương pháp phổ biến:**

**Netcat:** ví dụ lấy header HTTP:  
echo -e "HEAD / HTTP/1.0\r\n\r\n" | nc <target> 80

* **Telnet / nc:** mở kết nối tới port (ví dụ 25, 21) và đọc banner reply.
* **Curl:** curl -I http://<target> để lấy HTTP headers.
* **Ncat (ncat) với TLS:** ncat --ssl <target> 443 khi dịch vụ mã hóa.
* **Giới hạn:** nhiều dịch vụ che giấu hoặc làm giả banner; do đó banner nên được kết hợp với kết quả fingerprinting khác (nmap -sV, banner + response patterns).

### 2.2.3 Port scanning

* **Mục đích:** xác định port nào đang lắng nghe dịch vụ, từ đó xác định dịch vụ có thể truy cập.
* **Trạng thái port:** open, closed, filtered (không phản hồi / bị tường lửa lọc).
* **Các phương pháp quét chính:**
  + **TCP Connect Scan (full connect):** hoàn tất handshake; đáng tin cậy nhưng dễ bị phát hiện.
  + **SYN Scan (half-open):** gửi SYN, chờ SYN/ACK và gửi RST để không hoàn tất; nhanh và stealthier.
  + **UDP Scan:** gửi UDP packet và chờ ICMP unreachable; chậm và dễ false-positive.
  + **Các kiểu khác (FIN/NULL/Xmas):** lợi dụng cách TCP stack trả lời để suy luận; hiệu quả với một số OS nhưng dễ bị phòng thủ hiện đại chặn.

### 2.2.4 Công cụ điển hình và ví dụ lệnh

**Host discovery (LAN):**

* arping -c 2 192.168.56.1
* arp-scan 192.168.56.0/24

**Quick port check (netcat):**nc -z -v -w 1 192.168.56.102 22

* (-z: zero-I/O scan; -v: verbose; -w 1: timeout 1s)

**Banner grabbing (netcat):**echo -e "HEAD / HTTP/1.0\r\n\r\n" | nc 192.168.56.102 80

**Port scan đầy đủ (nmap):**sudo nmap -sS -sV 192.168.56.102

* (-sS: SYN scan; -sV: service/version detection)

**Bắt gói để phân tích (tcpdump):**sudo tcpdump -i eth0 host 192.168.56.102 -w recon\_capture.pcap

### 2.2.5 Quy trình recon gợi ý để thực nghiệm

1. Thiết lập môi trường lab (Kali attacker, victim VM, host-only/isolated network).
2. Passive recon (nếu có): thu thập thông tin công khai, capture traffic nếu được phép.
3. Host discovery trong LAN (ARP scan).
4. Port scanning trên host được phát hiện (dùng netcat + nmap để so sánh).
5. Banner grabbing cho các port mở; lưu banner và timestamp.
6. Capture traffic (tcpdump/Wireshark) để xác thực kết quả.
7. Tổng hợp kết quả, ghi chú false-positive/negative và đề xuất phương án kiểm chứng.

### 2.2.6 Hạn chế, rủi ro và lưu ý an toàn

* **False positives/negatives** thường xuất hiện, nhất là với UDP scan và các host sử dụng tường lửa. Luôn xác thực bằng nhiều công cụ.
* **Active recon để lại dấu vết** trên IDS/IPS; có thể kích hoạt cảnh báo bảo mật.
* **Pháp lý & đạo đức:** toàn bộ thí nghiệm phải có sự đồng ý rõ ràng; trong báo cáo cần nêu rõ phạm vi và cam kết an toàn.
* **Ghi chép đầy đủ:** lưu tất cả command, kết quả, pcap, timestamp và phiên bản công cụ để báo cáo có thể tái lập.

## 2.3 So sánh Netcat với các công cụ tương tự

Trong lĩnh vực kỹ thuật mạng, việc lựa chọn và sử dụng công cụ phù hợp để thực hiện các nhiệm vụ phân tích, kiểm tra và khai thác hệ thống mạng là rất quan trọng. Việc so sánh chi tiết Netcat với các tiện ích phổ biến khác như Ncat, Socat và hping3 sẽ cung cấp một cái nhìn toàn diện về điểm mạnh, điểm yếu và các trường hợp sử dụng tối ưu cho từng công cụ. Phần này sẽ tập trung vào việc phân tích các tính năng kỹ thuật cốt lõi để đưa ra sự khác biệt rõ ràng nhất.

Để có một cái nhìn tổng quan và dễ so sánh, chúng ta sẽ đánh giá các công cụ dựa trên những tính năng quan trọng sau đây thông qua bảng so sánh.

Bảng 3 Bảng so sánh Netcat với các tiện ích phổ biến khác như Ncat, Socat và hping3

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Tính năng** | **Netcat (truyền thống)** | **Ncat (Nmap's Netcat)** | **Socat** | **hping3** |
| **Mục đích chính** | Đọc/ghi dữ liệu TCP/UDP cơ bản. | Netcat cải tiến, bảo mật, nhiều tính năng [2]. | Chuyển tiếp (relay) luồng I/O 2 chiều. | Tạo/phân tích gói tin tùy chỉnh. |
| **Hỗ trợ giao thức** | TCP, UDP. | TCP, UDP, SCTP. | TCP, UDP, SCTP, Unix Sockets, Pipes... | TCP, UDP, ICMP, RAW IP. |
| **Hỗ trợ IPv6** | Không. | **Có.** | **Có.** | **Có.** |
| **Mã hóa (SSL/TLS)** | Không. | **Có** (Client & Server). | **Có** (Client & Server). | Không. |
| **Hỗ trợ Proxy** | Hạn chế. | **Có** (SOCKS/HTTP). | **Có** (SOCKS/HTTP). | Hạn chế. |
| **Kiểm soát truy cập** | Không. | **Có** (Allow/Deny IP). | **Có** (TCP wrappers). | Không áp dụng. |
| **Chế độ Broker** | Không. | **Có** (Kết nối nhiều client). | Không. | Không. |
| **Khả năng Scripting** | Tốt (đơn giản). | Rất tốt (tương thích ngược nc). | Mạnh mẽ (cú pháp phức tạp). | Mạnh mẽ (kiểm thử mạng). |
| **Tùy chỉnh gói tin** | Không. | Không. | Không. | **Tính năng cốt lõi** (Full header). |

* Từ Bảng 2.3, có thể thấy rằng mặc dù các công cụ này thường được nhóm chung, chúng phục vụ các mục đích rất khác nhau. Phần này sẽ phân tích sâu hơn về các khác biệt kỹ thuật và trường hợp sử dụng cụ thể của từng công cụ so với Netcat.

### 2.3.1. Netcat (nc) so với Ncat (ncat)

Ncat là phiên bản Netcat được phát triển bởi dự án Nmap, với mục tiêu trở thành một bản nâng cấp hiện đại và an toàn hơn . Sự khác biệt này là rõ ràng nhất:

* **Bảo mật và Mã hóa**: Đây là nâng cấp quan trọng nhất. Netcat truyền thống gửi toàn bộ dữ liệu dưới dạng văn bản thuần (plaintext), gây rủi ro an ninh nghiêm trọng. Ncat tích hợp hỗ trợ SSL/TLS, cho phép cả máy chủ và máy khách thiết lập các kênh liên lạc được mã hóa, đảm bảo tính bí mật và toàn vẹn của dữ liệu.
* **Hỗ trợ Mạng Hiện đại**: Ncat hỗ trợ đầy đủ IPv6 và khả năng kết nối qua các proxy phổ biến (SOCKS4, SOCKS5, và HTTP). Netcat truyền thống thường bị giới hạn ở IPv4 và không có hỗ trợ proxy tích hợp.
* **Quản lý Kết nối**: Ncat giới thiệu các tính năng quản lý tiên tiến. Chế độ "Broker" cho phép Ncat hoạt động như một máy chủ trung gian, kết nối nhiều máy khách với nhau. Thêm vào đó, Ncat có thể thực thi kiểm soát truy cập đơn giản thông qua các tùy chọn --allow và --deny để lọc kết nối dựa trên địa chỉ IP, một tính năng mà Netcat gốc không có.
* Ncat là sự thay thế vượt trội cho Netcat truyền thống trong hầu hết mọi kịch bản hiện đại, đặc biệt là khi bảo mật và khả năng tương thích mạng (IPv6, proxy) là yêu cầu bắt buộc.

### 2.3.2. Netcat/Ncat so với Socat

Nếu Ncat là một sự *nâng cấp* của Netcat, thì Socat (Socket CAT) là một công cụ có triết lý thiết kế *tổng quát hóa* .

* **Triết lý hoạt động**: Netcat/Ncat được thiết kế để làm việc chủ yếu với các socket mạng (TCP/UDP). Socat được thiết kế để chuyển tiếp (relay) dữ liệu hai chiều giữa hai điểm cuối (addresses) bất kỳ.
* **Tính linh hoạt của điểm cuối**: Điểm mạnh tuyệt đối của Socat nằm ở đây. Một "address" trong Socat không chỉ là một socket TCP/UDP, mà có thể là một file, một pipe (ống dẫn), một thiết bị đầu cuối (TTY), một Unix socket, hay thậm chí là một tiến trình con. Điều này cho phép Socat thực hiện các tác vụ chuyển tiếp cực kỳ phức tạp, ví dụ:
  + Chuyển tiếp một cổng serial (thiết bị vật lý) sang một kết nối TCP.
  + Tạo một Unix socket để lắng nghe và chuyển tiếp dữ liệu đến một máy chủ TCP từ xa.
* **Độ phức tạp**: Sự linh hoạt này đi kèm với một cú pháp phức tạp hơn đáng kể. Thay vì các cờ đơn giản (-l, -p), Socat yêun cầu người dùng định nghĩa rõ ràng cả hai điểm cuối và các tham số của chúng.
* Ncat phù hợp cho các tác vụ client-server, truyền file hoặc quét cổng. Socat là công cụ chuyên dụng cho các tác vụ *chuyển tiếp* (relay/forwarding) phức tạp, đặc biệt khi cần liên kết các loại I/O khác nhau (không chỉ mạng).

### 2.3.3. Netcat/Ncat so với hping3

Hping3 hoạt động ở một tầng thấp hơn trong mô hình mạng và có mục đích hoàn toàn khác [4].

* **Mục đích cốt lõi**: Netcat/Ncat được sử dụng để *truyền dữ liệu* qua một kết nối đã được thiết lập (Tầng 4 - Transport). Ngược lại, hping3 được thiết kế để *tạo ra (crafting) và phân tích* các gói tin (packet) riêng lẻ ở Tầng 3 (Network) và Tầng 4 (Transport).
* **Khả năng tùy chỉnh**: hping3 cho phép người dùng kiểm soát toàn bộ các trường trong header của gói tin. Ví dụ, người dùng có thể tạo một gói tin TCP với các cờ (flag) tùy chỉnh (SYN, ACK, RST, FIN...), thiết lập Time-To-Live (TTL), hoặc thay đổi kích thước cửa sổ (window size) .
* **Trường hợp sử dụng**: hping3 không dùng để truyền file hay chat. Nó là công cụ chuyên dụng để:
  + Kiểm thử và phân tích quy tắc của tường lửa (Firewall testing).
  + Quét cổng nâng cao (ví dụ: quét ACK, quét FIN).
  + Chẩn đoán các vấn đề mạng ở mức độ gói tin.
* Hping3 không phải là công cụ thay thế cho Netcat mà là một công cụ bổ trợ. Chúng thường được sử dụng cùng nhau: hping3 để thăm dò mạng và tường lửa, và Ncat để tương tác với các dịch vụ được phát hiện là đang mở.

## 2.4 Vấn đề an toàn, pháp lý và đạo đức khi dùng Netcat

Netcat, do bản chất linh hoạt của nó, là một công cụ có tính "hai mặt" (dual-use). Nó vừa là một tiện ích vô giá cho quản trị viên hệ thống, vừa là một vũ khí hiệu quả trong kho công cụ của tin tặc. Phần này sẽ phân tích các vấn đề về an toàn (phát hiện, mã hóa), pháp lý và đạo đức liên quan đến việc sử dụng công cụ này.

### 2.4.1 Vấn đề an toàn: Phát hiện, Dấu vết và Mã hóa

Một trong những đặc điểm thiết kế nguyên thủy của Netcat là sự đơn giản. Tuy nhiên, chính sự đơn giản này lại dẫn đến các vấn đề an toàn nghiêm trọng trong bối cảnh mạng hiện đại.

* **Truyền dữ liệu dạng văn bản thuần (Plaintext):** Phiên bản Netcat truyền thống không tích hợp bất kỳ cơ chế mã hóa nào. Toàn bộ dữ liệu, bao gồm cả các lệnh được thực thi trong một reverse shell hoặc nội dung file được truyền, đều được gửi dưới dạng văn bản thuần. Điều này khiến nó:
  + **Dễ bị phát hiện:** Bất kỳ Hệ thống phát hiện xâm nhập (IDS/IPS) hoặc công cụ phân tích gói tin (như Wireshark) nào cũng có thể dễ dàng bắt và đọc được nội dung .
  + **Dễ bị tấn công Xen giữa (Man-in-the-Middle):** Kẻ tấn công có thể chặn và thay đổi dữ liệu đang được truyền.
* **Dấu vết và Phát hiện trên máy chủ (Host-based Detection):**
  + **Lịch sử dòng lệnh:** Việc sử dụng nc thường để lại dấu vết rõ ràng trong lịch sử shell của người dùng (ví dụ: .bash\_history), tiết lộ lệnh đã được thực thi.
  + **Tiến trình (Processes):** Một tiến trình nc chạy với các cờ đáng ngờ (ví dụ: nc -l -p [port] -e /bin/bash) rất dễ bị phát hiện bởi các công cụ giám sát hệ thống hoặc các giải pháp EDR (Endpoint Detection and Response).
  + **Kết nối mạng:** Một kết nối TCP hoặc UDP tới một cổng không chuẩn (non-standard port) từ một tiến trình nc là một chỉ dấu bất thường và thường bị các tường lửa máy chủ (host-based firewalls) ghi lại (log).
* **Giải pháp và Biện pháp đối phó (Countermeasures):**
  + **Về phía người dùng/tấn công:** Để giải quyết vấn đề thiếu mã hóa, các biến thể an toàn hơn như **Ncat** (với tùy chọn --ssl) [1] hoặc sử dụng kỹ thuật đường hầm (tunneling) qua SSH đã trở nên phổ biến. Một số tin tặc còn sử dụng các phiên bản Netcat đã được làm mờ (obfuscated) hoặc đổi tên file thực thi (nc thành svc) để tránh bị phát hiện dựa trên tên tiến trình.
  + **Về phía phòng thủ:** Các nhà phân tích an ninh mạng không chỉ tìm kiếm nc trong log. Họ tìm kiếm các *hành vi* bất thường: một tiến trình không xác định lắng nghe trên một cổng, hoặc một tiến trình shell (/bin/bash) được sinh ra bởi một kết nối mạng.

### 2.4.2 Vấn đề Pháp lý và Quy định

Tính hợp pháp của việc sử dụng Netcat hoàn toàn phụ thuộc vào **sự cho phép (authorization)** và **ý định (intent)** của người dùng.

* **Sử dụng hợp pháp (Legal Use):**
  + Quản trị viên hệ thống sử dụng Netcat để gỡ lỗi các dịch vụ mạng.
  + Kỹ sư mạng kiểm tra kết nối và quy tắc tường lửa.
  + Chuyên gia kiểm thử xâm nhập (Pentester) sử dụng Netcat trong phạm vi (scope) của một hợp đồng đã ký kết rõ ràng với khách hàng.
* **Sử dụng bất hợp pháp (Illegal Use):**
  + **Quét cổng trái phép:** Mặc dù Netcat có thể quét cổng, việc quét cổng một hệ thống mà không có sự cho phép có thể bị coi là hành vi xâm nhập và vi phạm pháp luật ở nhiều quốc gia.
  + **Cài đặt Backdoor (Reverse/Bind Shells):** Sử dụng Netcat để tạo một cửa hậu (-e /bin/bash) trên hệ thống của nạn nhân là một hành vi xâm nhập máy tính rõ ràng. Tại Việt Nam, hành vi này có thể cấu thành tội "Xâm nhập trái phép vào mạng máy tính, mạng viễn thông hoặc phương tiện điện tử của người khác" (Điều 289, Bộ luật Hình sự 2015) .
  + **Trích xuất dữ liệu (Data Exfiltration):** Sử dụng Netcat để gửi các tệp tin nhạy cảm từ máy nạn nhân ra bên ngoài là hành vi trộm cắp dữ liệu.
* Bản thân Netcat chỉ là một công cụ. Việc sở hữu nó không phải là bất hợp pháp, nhưng việc sử dụng nó để truy cập hoặc can thiệp vào các hệ thống không thuộc quyền sở hữu hoặc quản lý của mình là vi phạm pháp luật.

### 2.4.3 Vấn đề Đạo đức

Vấn đề đạo đức nảy sinh ngay cả khi hành vi có thể chưa bị truy tố về mặt pháp lý.

* **Quyền riêng tư (Privacy):** Sử dụng Netcat để "lắng nghe" (listen) trên một cổng dịch vụ của đồng nghiệp để xem họ đang làm gì, ngay cả khi không có ý đồ xấu, đã là một sự vi phạm quyền riêng tư và đạo đức nghề nghiệp.
* **Vô trách nhiệm (Recklessness):** Một quản trị viên hệ thống sử dụng Netcat (phiên bản không mã hóa) để truyền các tệp tin chứa mật khẩu hoặc dữ liệu nhạy cảm qua mạng công ty là một hành vi vô trách nhiệm, đặt tổ chức vào tình thế rủi ro, ngay cả khi mục đích là công việc.
* **Vượt quá phạm vi (Scope Creep):** Trong một cuộc kiểm thử xâm nhập, nếu chuyên gia phát hiện ra một lỗ hổng và sử dụng Netcat để truy cập vào máy chủ, nhưng sau đó tiếp tục "khám phá" sang các máy chủ khác không nằm trong phạm vi hợp đồng, hành vi đó là phi đạo đức.

**Kết luận:**

Netcat là một công cụ mạnh mẽ, nhưng sức mạnh của nó đòi hỏi người dùng phải có trách nhiệm cao. Việc thiếu các tính năng an toàn tích hợp (như mã hóa) trong phiên bản gốc khiến nó trở thành một lựa" chọn rủi ro cho các tác vụ hợp pháp trong môi trường nhạy cảm. Về mặt pháp lý và đạo đức, ranh giới được xác định rõ ràng bởi sự cho phép: mọi hành vi sử dụng Netcat để truy cập, quét hoặc lấy dữ liệu từ một hệ thống mà không có sự đồng ý rõ ràng của chủ sở hữu đều là vi phạm.

# 

# CHƯƠNG 3: TRIỂN KHAI CÔNG CỤ NETCAT

## 3.1 Khảo sát host và banner grabbing

### 3.1.1 Mục tiêu

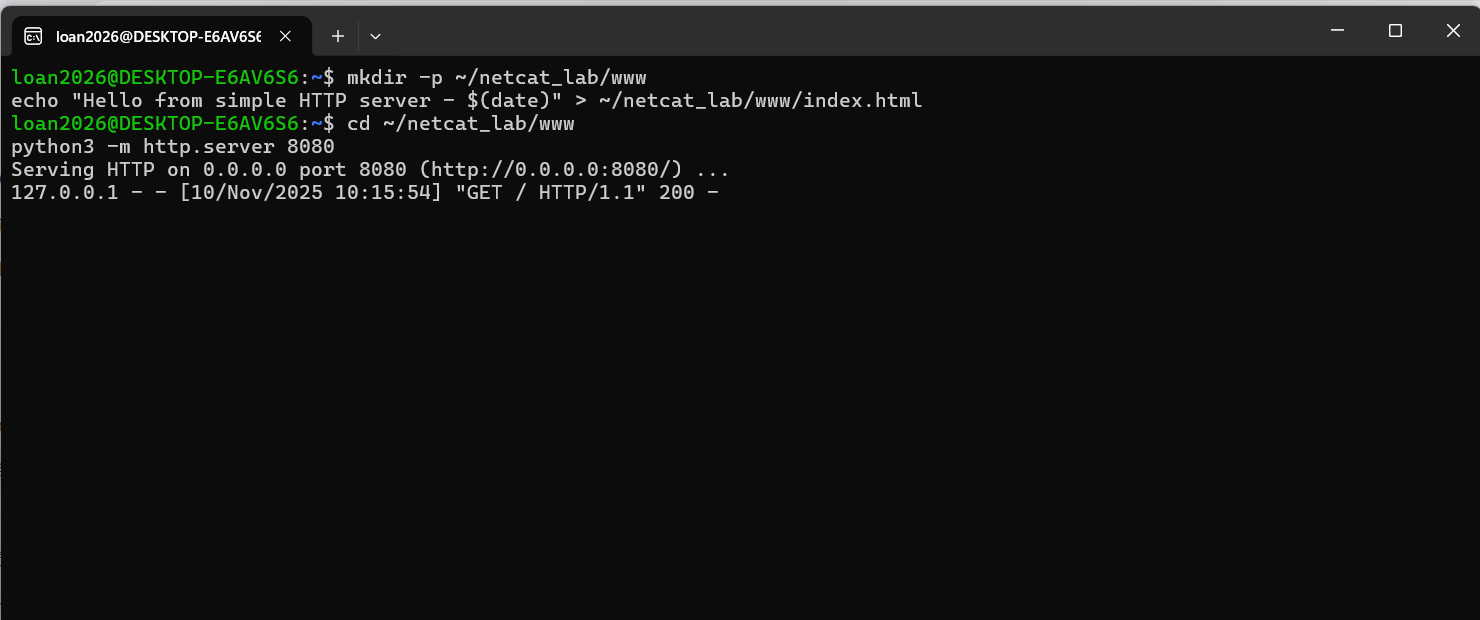
Thực hiện thu thập thông tin ban đầu (banner grabbing) để xác định dịch vụ và phiên bản chạy trên cổng mục tiêu.

### 3.1.2 Công cụ sử dụng

* Ncat/Netcat
* Python simple HTTP server
* Wireshark / tshark để ghi pcap

### 3.1.3 Các bước thực hiện

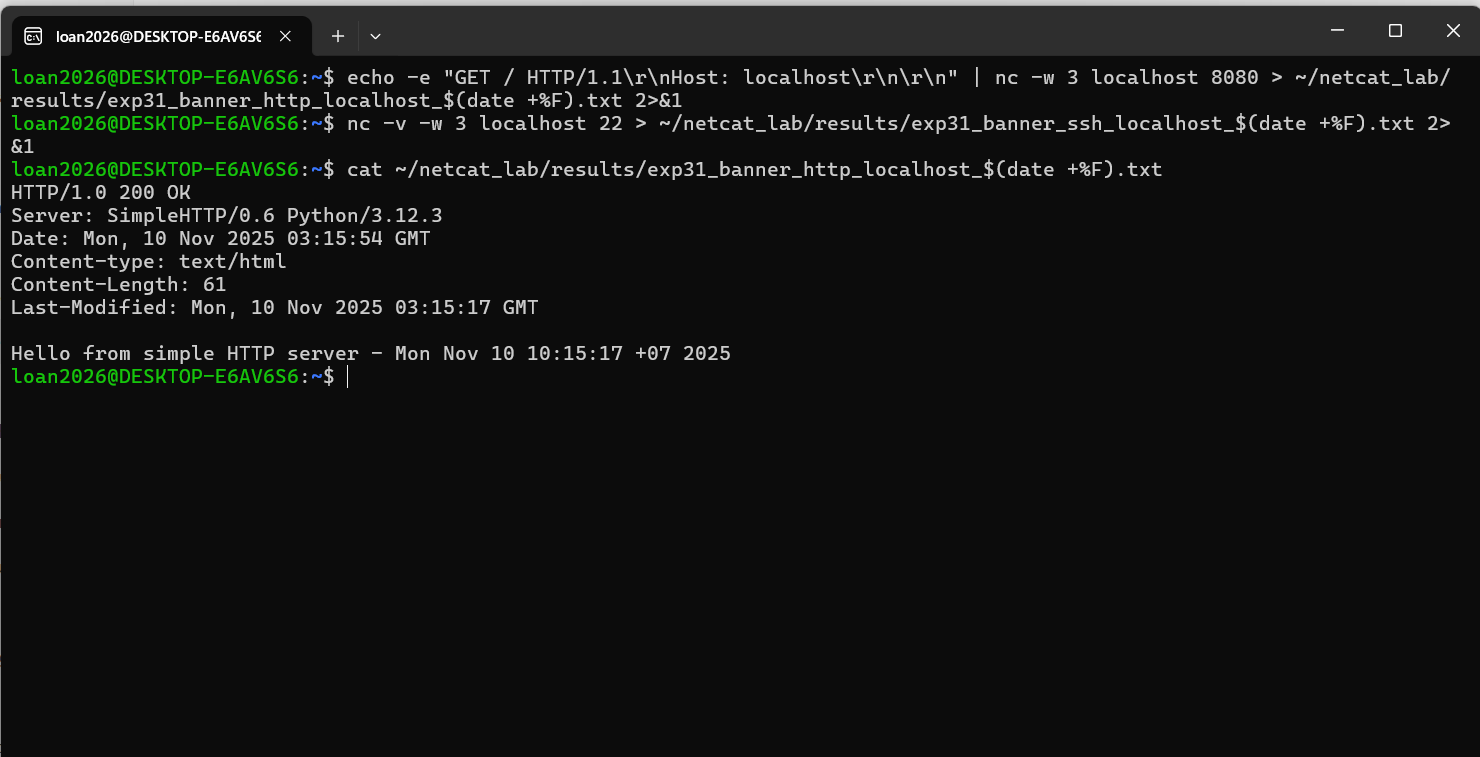
**Bước 1:** Khởi HTTP test server



Hình 1. Khởi tạo test sever

* *mkdir -p ~/netcat\_lab/www:* tạo thư mục *~/netcat\_lab/www. -p* đảm bảo tạo cả thư mục cha nếu chưa có, không báo lỗi nếu đã tồn tại.
* *echo "Hello from simple HTTP server - $(date)" > ~/netcat\_lab/www/index.html:* lệnh *echo* in chuỗi ra stdout; *$(date)* được shell thay bằng thời gian hiện tại. Dấu > chuyển hướng output vào file *index.html* (ghi đè file nếu tồn tại).
* *python3 -m http.server 8080*: khởi một web server nhỏ của Python và lắng nghe trên cổng **8080**.

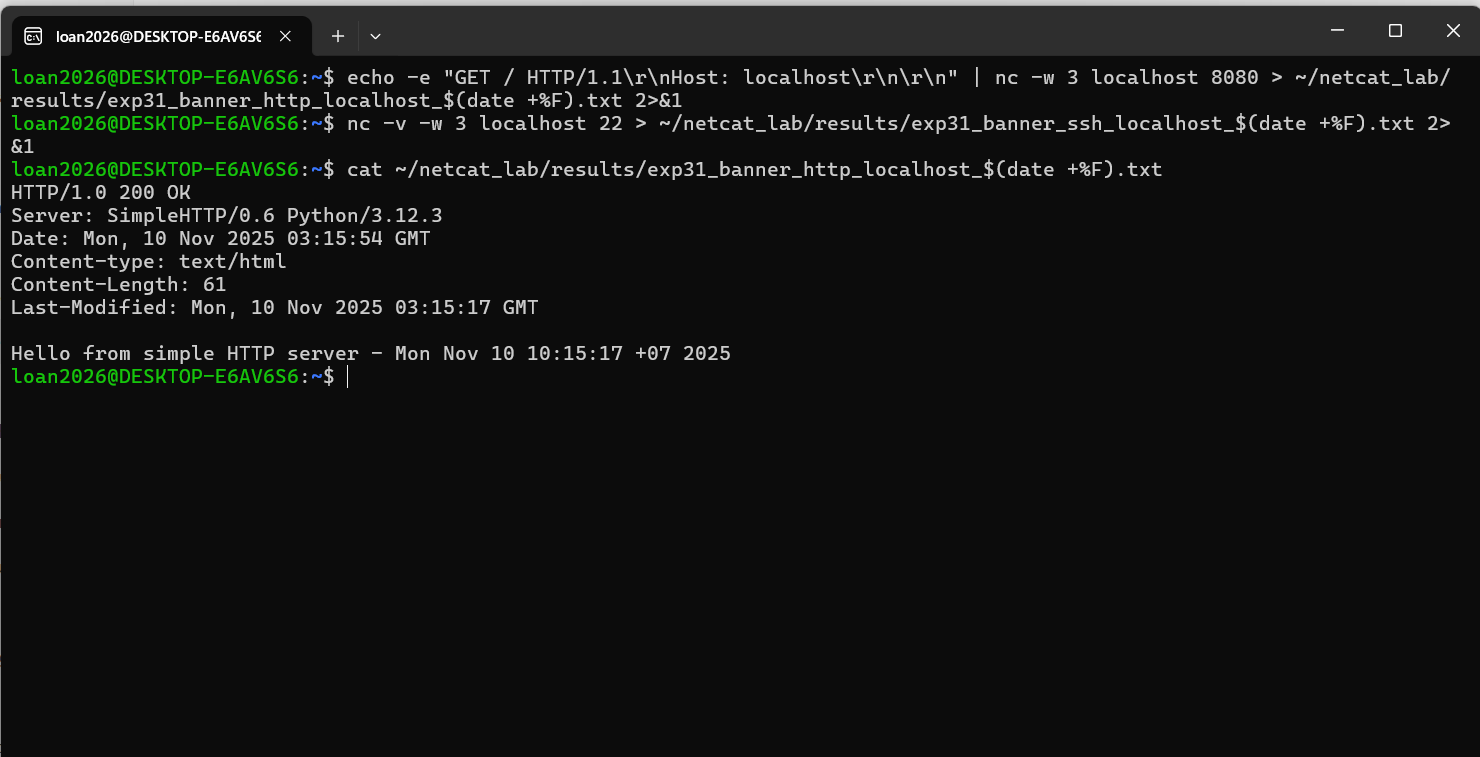
**Bước 2:** Banner grabbing bằng netcat/ncat



Hình 2. Banner grabbing

* Tạo yêu cầu HTTP thô *(GET / ...*) rồi gửi vào *nc* nối tới *localhost:8080.*
* *-w 3* = timeout 3 giây (tránh chờ lâu).
* Toàn bộ output (stdout + stderr) được ghi vào file *~/netcat\_lab/results/exp31\_banner\_http\_localhost\_YYYY-MM-DD.txt.*
* Kết quả file chứa header HTTP và body (nội dung trang).
* Kết nối tới cổng 22 (SSH). -v bật verbose để hiển thị thông tin kết nối.
* Nhiều SSH server trả banner ngay khi kết nối (ví dụ *SSH-2.0-OpenSSH\_8.4*), nên lệnh này thường thu được banner SSH.

### 3.1.4 Kết quả



Hình 3. Kết quả thu thập banner

* Mục tiêu của phần này là thu thập banner từ các dịch vụ chạy trên host mục tiêu để xác định loại dịch vụ và phiên bản. Thí nghiệm được thực hiện trên localhost (HTTP: cổng 8080) bằng cách gửi yêu cầu HTTP thủ công qua ncat. Kết quả thu được cho thấy server trả về header Server: SimpleHTTP/..., cho phép xác định dịch vụ HTTP đang chạy.

## 3.2 Kiểm tra trạng thái cổng (Port check / Connect scan)

### 3.2.1 Mục tiêu

Mục tiêu của phần thực hành này là kiểm tra trạng thái của các cổng mạng (port) trên máy Kali Linux bằng kỹ thuật Connect Scan sử dụng công cụ Netcat.  
 Nội dung bao gồm:

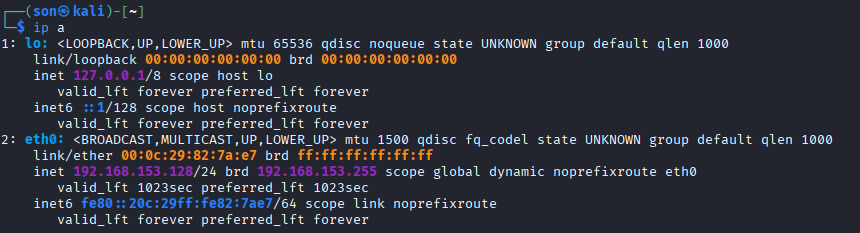
* Xác định địa chỉ IP của máy Kali Linux và máy chủ Host (Windows).
* Sử dụng Netcat để kiểm tra trạng thái của một cổng TCP (mở – đóng – từ chối kết nối).
* Kích hoạt dịch vụ SSH trên Kali để tạo một cổng đang mở nhằm phục vụ quá trình kiểm tra.
* Quan sát sự thay đổi trạng thái port trước và sau khi dịch vụ được kích hoạt.
* Hiểu được ý nghĩa của các phản hồi từ netcat trong quá trình Connect Scan.

### 3.2.2 Các bước thực hiện

**Bước 1**: Kiểm tra địa chỉ IP của máy Kali Linux

Sử dụng lệnh: ip a

Kết quả cho thấy địa chỉ IP của máy Kali là:192.168.153.128



Hình 4. Kiểm tra dịa chỉ IP của máy Kali Linux

**Bước 2:** Kiểm tra địa chỉ IP của máy Windows Host

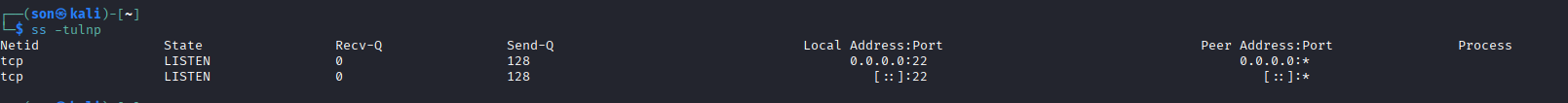
Chạy cmd trên Window và chạy lệnh ipconfig sẽ cho thấy ip địa chỉ IP của adapter VMware NAT là:192.168.153.1



Hình 5. Kiểm tra địa chỉ IP của máy Window

**Bước 3 :** Kiểm tra các port đang lắng nghe trên Kali

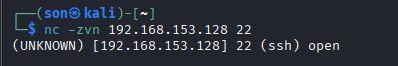
Thực hiện lệnh ss -tulnp trên kali linux sẽ cho thấy các cổng đang được lắng nghe



Hình 6. Kiểm tra các port lắng

Bước 4: Kiểm tra trạng thái port ngẫu nhiên trên Kali

Sử dụng lệnh Connect Scan:nc -zvn 192.168.153.128 22, ở đây ta kiểm tra cổng 22 với ssh, sau khi chạy thì Netcat phản hồi trạng thái Open, điều này thể hiện rằng port 22 đang mở



Hình 7. Kiểm tra port ngẫu nhiên

### 3.2.3 Kết quả

**Kết quả thu được:**

* **Cổng Mở (Open Port):** Khi thực hiện kết nối Netcat đến một cổng đang hoạt động , Netcat thiết lập kết nối thành công và giữ phiên làm việc. Phản hồi cho thấy cơ chế bắt tay ba bước (three-way handshake) của TCP đã hoàn tất.
* **Tính Hiệu quả:** Netcat, đặc biệt khi sử dụng cờ -z (zero-I/O mode)chứng minh là một công cụ **nhanh chóng và nhẹ** để xác định trạng thái mở/đóng của một cổng đơn lẻ hoặc một dải cổng nhỏ. Nó thực hiện quét kết nối TCP đầy đủ (full connect scan), đáng tin cậy hơn các phương pháp quét 'nửa mở' (half-open scan) nhưng lại **dễ bị các hệ thống phòng thủ phát hiện và ghi log** vì hoàn tất kết nối.

**Phân tích kết quả**

Lệnh kiểm tra nc -zvn 192.168.153.128 22 trả về kết quả “22 (ssh) open”, cho thấy port 22 trên máy Kali đang mở và dịch vụ SSH hoạt động bình thường. Kết quả này phù hợp với lệnh ss -tulnp, nơi tiến trình sshd xuất hiện ở trạng thái LISTEN.   
Trước khi bật SSH, Netcat báo “connection refused”, chứng tỏ port đóng; sau khi kích hoạt dịch vụ, port chuyển sang “open”, thể hiện cơ chế Connect Scan hoạt động chính xác. Thông báo “(UNKNOWN)” chỉ do không có hostname cho IP và không ảnh hưởng đến việc xác định trạng thái cổng.

## 3.3 Lắng nghe và ghi nhận kết nối (Listen / Logging)

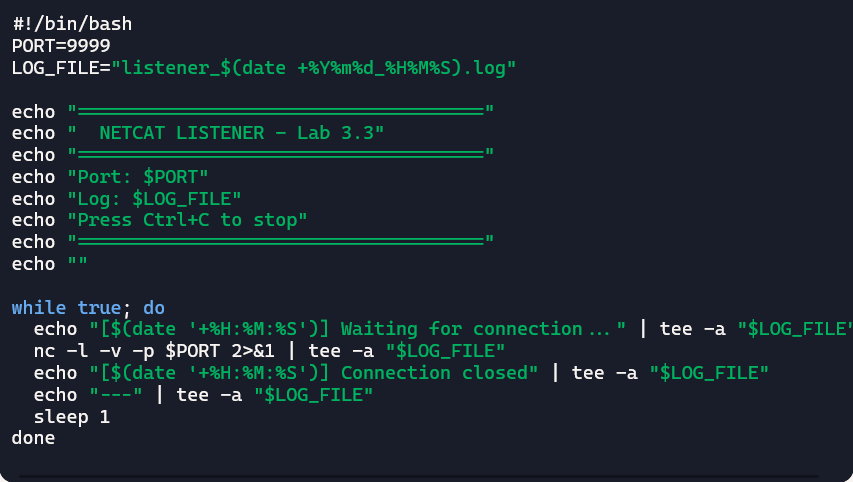
### 3.3.1 Mục tiêu

Phần thực nghiệm này nhằm minh họa khả năng của công cụ Netcat khi hoạt động ở chế độ máy chủ (listener mode). Mục tiêu cụ thể bao gồm:

* Thiết lập một listener sử dụng Netcat để chờ và ghi nhận các kết nối đến từ phía client.
* Ghi lại nội dung truyền nhận giữa hai điểm kết nối vào tệp log nhằm phục vụ phân tích.
* Đánh giá khả năng giám sát kết nối mạng của Netcat trong môi trường kiểm thử.

### 3.3.2 Các bước thực hiện

**Bước 1:** Trên máy Kali Linux đóng vai trò máy chủ, tạo một tập tin shell script có tên listener\_loop.sh với nội dung như sau:



Hình 8. Tạo tập tin shell script

Sau khi tạo xong, cấp quyền thực thi cho script bằng lệnh: chmod +x listener\_loop.sh

Khởi chạy listener: ./listener\_loop.sh

**Bước 2**: Gửi dữ liệu từ máy cilent

Trên máy Kali thứ hai (hoặc cùng máy nếu kiểm thử nội bộ), thực hiện gửi dữ liệu đến listener bằng lệnh: echo "Test message from client - $(date)" | nc <IP của máy chủ> 9999

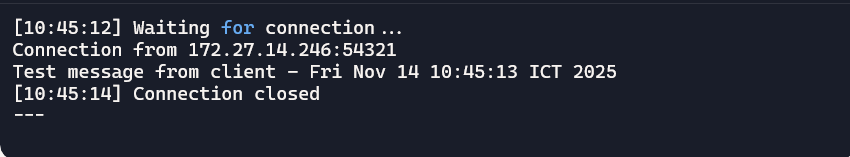
**Bước 3**: Kiểm tra và phân tích log

Sau khi nhận kết nối, máy chủ sẽ ghi lại nội dung vào tệp log có tên dạng Sau khi nhận kết nối, máy chủ sẽ ghi lại nội dung vào tệp log có tên dạng listener\_YYYYMMDD\_HHMMSS.log. Có thể kiểm tra bằng lệnh: cat listener\_\*.lolog

### 3.3.3 Kết quả

Kết quả thực nghiệm cho thấy Netcat hoạt động ổn định ở chế độ listener. Mỗi lần kết nối được ghi nhận đầy đủ thông tin gồm:

* Thời điểm bắt đầu và kết thúc kết nối.
* Địa chỉ IP và cổng của client.
* Nội dung dữ liệu được truyền từ client.



Hình 9. Kết quả lắng và ghi nhận kết nối ở chế độ listener

**Phân tích:**

* Netcat cho phép giám sát kết nối mạng một cách đơn giản và hiệu quả trong môi trường kiểm thử.
* Tệp log được tạo tự động và có định dạng rõ ràng, thuận tiện cho việc phân tích và tái hiện quá trình.
* Tuy nhiên, trong môi trường thực tế, cần bổ sung các cơ chế bảo mật và xác thực để tránh bị lạm dụng.

## 3.4 Truyền file đơn giản qua Netcat (Sender/Receiver)

### 3.4.1 Mục tiêu

Mục tiêu của phần thực nghiệm này là minh họa khả năng truyền tải dữ liệu giữa hai máy tính thông qua công cụ Netcat trong môi trường mạng nội bộ. Cụ thể:

* Thiết lập kết nối giữa máy gửi và máy nhận sử dụng Netcat.
* Thực hiện truyền một tệp tin đơn giản từ máy gửi sang máy nhận.
* Ghi nhận và phân tích kết quả truyền tải để đánh giá tính hiệu quả và độ tin cậy của Netcat trong vai trò công cụ hỗ trợ truyền dữ liệu.

### 3.4.2 Các bước thực hiện

**Bước 1**: Chuẩn bị tệp tin cần truyền

Trên máy gửi (Sender), tạo một tệp tin ví dụ:



Hình 10. Chuẩn bị tệp tin cho máy gửi

**Bước 2**: Thiết lập listener trên máy nhận (Receiver)

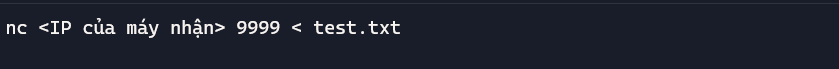
Trên máy nhận, mở terminal và chạy lệnh sau để lắng nghe kết nối và ghi dữ liệu nhận được vào tệp:



Hình 11. Thiết lập listener trên máy nhận

**Bước 3**: Gửi tệp tin từ máy gửi

Trên máy gửi, thực hiện lệnh sau để gửi tệp test.txt đến máy nhận:



Hình 12. Gửi tệp tin từ máy gửi

**Bước 4:** Kiểm tra kết quả truyền tải

Trên máy nhận, sau khi quá trình truyền kết thúc, kiểm tra nội dung tệp nhận được:



Hình 13. Kết quả truyền tải

### 3.4.3 Kết quả

**Kết quả thực nghiệm:**

* Tệp tin test.txt được truyền thành công từ máy gửi sang máy nhận.
* Nội dung tệp received\_test.txt trên máy nhận khớp hoàn toàn với nội dung gốc.
* Quá trình truyền diễn ra nhanh chóng và không gặp lỗi trong môi trường mạng nội bộ.

**Phân tích:**

* Netcat cho thấy khả năng truyền dữ liệu hiệu quả trong môi trường kiểm thử, đặc biệt phù hợp với các tệp văn bản nhỏ.
* Việc truyền dữ liệu không sử dụng giao thức mã hóa, do đó không đảm bảo tính bảo mật nếu triển khai trong môi trường thực tế.
* Netcat không có cơ chế xác thực hoặc kiểm tra toàn vẹn dữ liệu, vì vậy cần thận trọng khi sử dụng trong các hệ thống yêu cầu độ tin cậy cao.

**Kết luận:**

Thực nghiệm truyền file qua Netcat đã minh họa rõ ràng cách thiết lập kết nối và chuyển dữ liệu giữa hai máy. Đây là một chức năng hữu ích của Netcat trong các tình huống kiểm thử, tuy nhiên cần bổ sung các biện pháp bảo mật nếu áp dụng trong môi trường sản xuất.

## 3.5 Kết hợp Netcat với công cụ khác (pipeline / xử lý)

### 3.5.1 Mục tiêu

Mục tiêu của phần này là minh họa khả năng kết hợp Netcat (nc) với các công cụ xử lý dòng dữ liệu như sed, grep thông qua cơ chế **pipeline (|)**.  
Việc kết hợp này giúp:

* Tự động hóa phân tích dữ liệu phản hồi từ máy chủ.
* Trích xuất chính xác các thông tin quan trọng từ header HTTP (ví dụ: Content-Type, Server, Content-Length).
* Giảm thao tác thủ công khi thực hiện thu thập thông tin (reconnaissance) hoặc banner grabbing.

### 3.5.2 Các bước thực hiện

Quy trình được thực hiện trên môi trường dòng lệnh (Terminal) bằng cách sử dụng một lệnh duy nhất kết nối ba tiện ích qua pipeline.

**Bước 1**: Chuẩn bị Lệnh Pipeline

Tạo và gửi yêu cầu **HTTP HEAD** tới máy chủ web (ví dụ: example.com trên cổng 80). Yêu cầu HEAD chỉ trả về các tiêu đề (Headers) mà không cần nội dung trang (Body), giúp tối ưu hóa luồng dữ liệu.

**Bước 2**: Thực thi Lọc Dữ liệu

Chạy lệnh sau trong Terminal. Lệnh này chuyển đầu ra của printf làm đầu vào cho nc, và đầu ra của nc (tức là các Headers) làm đầu vào cho sed.

Bash

printf "HEAD / HTTP/1.1\r\nHost: example.com\r\nConnection: close\r\n\r\n" | nc example.com 80 | sed -n '/Content-Type:/p'

Bảng 4 Chức năng và thành phần lọc dữ liệu

|  |  |
| --- | --- |
| **Thành phần** | **Chức năng** |
| printf "..." | Tạo gói yêu cầu HTTP HEAD đúng chuẩn |
| nc example.com 80 | Thiết lập kết nối TCP tới cổng 80 của máy chủ và gửi yêu cầu |
| sed -n '/Content-Type:/p' | Lọc ra duy nhất dòng chứa “Content-Type:” từ toàn bộ HTTP header |

Dữ liệu di chuyển theo pipeline từ trái sang phải:

**printf → nc → sed → output**

### 3.5.3 Kết quả



Hình 14. Kết quả thực hiện

**Phân tích kết quả:**

* Hiệu suất Giao tiếp Mạng: Lệnh nc đã hoàn thành vai trò kết nối, gửi yêu cầu HEAD thành công và nhận lại tất cả các Tiêu đề (Headers) từ máy chủ.
* Minh họa Pipeline: Toàn bộ dữ liệu Headers thô đã được chuyển tiếp liền mạch qua ống (|) và không hề hiển thị lên màn hình. Điều này chứng minh rằng Netcat có thể đóng vai trò là nguồn dữ liệu mạng cho các công cụ xử lý khác.
* Trích xuất Chính xác (sed): Công cụ sed đã thực hiện chức năng lọc và cắt gọt dữ liệu nâng cao, chỉ giữ lại dòng thông tin cụ thể cần thiết (Content-Type), loại bỏ tất cả các tiêu đề dư thừa khác (như Date, Server, Connection, v.v.).

**Kết luận:** Thí nghiệm này chứng minh sự hiệu quả và linh hoạt của việc kết hợp Netcat với các công cụ xử lý luồng dữ liệu (pipeline) để tự động hóa việc trích xuất thông tin mục tiêu từ phản hồi mạng.

# 

# CHƯƠNG 4: TỔNG KẾT VÀ ĐÁNH GIÁ

## 4.1 Nhận xét về quá trình thực hiện

### 4. 1. 1. Thuận lợi

Trong quá trình nghiên cứu và thực nghiệm công cụ Netcat, nhóm đã gặp được một số thuận lợi đáng kể:

* **Về môi trường thực nghiệm:** Hệ điều hành Kali Linux đã tích hợp sẵn Netcat cùng các công cụ hỗ trợ như Wireshark, tcpdump và Python HTTP server, giúp việc thiết lập môi trường lab diễn ra nhanh chóng mà không cần cài đặt thêm phần mềm. Việc sử dụng máy ảo VMware với mạng NAT/Host-only cũng tạo điều kiện thuận lợi cho việc kiểm soát và cô lập môi trường thử nghiệm.
* **Về tài liệu và nguồn học liệu:** Netcat là công cụ phổ biến với lịch sử phát triển lâu dài, do đó có nhiều tài liệu hướng dẫn chi tiết từ cộng đồng an ninh mạng. Các trang man page và documentation chính thức cung cấp đầy đủ thông tin về cú pháp và tham số sử dụng.

**Về tính đơn giản của công cụ:** Cú pháp lệnh của Netcat tương đối trực quan và dễ hiểu, phù hợp cho việc học tập và thực hành các khái niệm cơ bản về giao thức mạng TCP/UDP. Khả năng kết hợp với các công cụ khác thông qua cơ chế pipeline giúp mở rộng phạm vi ứng dụng một cách linh hoạt.

### 4. 1. 2. Khó khăn

Bên cạnh những thuận lợi, nhóm cũng gặp phải một số khó khăn trong quá trình thực hiện:

* **Về sự khác biệt giữa các biến thể:** Như đã phân tích trong Chương 2, tồn tại nhiều phiên bản Netcat khác nhau (netcat-traditional, netcat-openbsd, ncat) với cú pháp và tham số không hoàn toàn thống nhất. Điều này đôi khi gây nhầm lẫn khi tham khảo tài liệu từ các nguồn khác nhau hoặc khi chuyển đổi giữa các hệ thống.
* **Về hạn chế trong quét cổng:** Netcat không được thiết kế chuyên biệt cho việc quét cổng hàng loạt, do đó tốc độ quét chậm hơn đáng kể so với các công cụ chuyên dụng như Nmap. Việc quét một dải IP lớn hoặc nhiều cổng đồng thời đòi hỏi phải viết script bổ sung.
* **Về vấn đề bảo mật kênh truyền:** Phiên bản Netcat truyền thống không hỗ trợ mã hóa, khiến việc thực nghiệm trong môi trường mạng thực tế tiềm ẩn rủi ro. Nhóm phải sử dụng Ncat với tùy chọn SSL hoặc giới hạn thực nghiệm trong mạng nội bộ cô lập.

**Về việc phân tích kết quả:** Netcat trả về dữ liệu thô mà không có cơ chế phân tích tự động, đòi hỏi người dùng phải có kiến thức nền tảng về các giao thức mạng để diễn giải kết quả banner grabbing hoặc phản hồi từ các dịch vụ.

### 4. 1. 3. Đánh giá kết quả thực nghiệm

Qua các thực nghiệm được trình bày trong Chương 3, nhóm đưa ra những đánh giá sau:

* **Về tính hiệu quả của Netcat trong thu thập thông tin:** Netcat đã chứng minh được khả năng thực hiện các tác vụ thu thập thông tin cơ bản một cách hiệu quả. Thực nghiệm banner grabbing (mục 3.1) cho thấy công cụ có thể thu thập thành công thông tin về phiên bản dịch vụ HTTP và SSH. Thực nghiệm kiểm tra trạng thái cổng (mục 3.2) xác nhận cơ chế Connect Scan hoạt động chính xác, phân biệt rõ ràng giữa cổng mở và cổng đóng.
* **Về độ tin cậy của kết quả:** Các kết quả thu được từ Netcat có độ tin cậy cao khi được xác minh chéo với các công cụ khác. Ví dụ, kết quả kiểm tra cổng 22 bằng Netcat khớp với thông tin từ lệnh ss -tulnp. Tuy nhiên, cần lưu ý rằng Netcat thực hiện quét kết nối đầy đủ (full connect scan), dễ bị phát hiện và ghi log bởi các hệ thống giám sát.
* **Về khả năng ứng dụng thực tế:** Thực nghiệm lắng nghe và ghi nhận kết nối (mục 3.3) cùng thực nghiệm truyền file (mục 3.4) cho thấy Netcat phù hợp cho các tác vụ kiểm thử trong môi trường lab có kiểm soát. Tuy nhiên, do thiếu cơ chế mã hóa và xác thực, việc triển khai trong môi trường sản xuất cần được cân nhắc kỹ lưỡng và bổ sung các biện pháp bảo mật phù hợp.
* **Về khả năng mở rộng và tích hợp:** Thực nghiệm kết hợp Netcat với pipeline (mục 3.5) chứng minh tính linh hoạt của công cụ khi tích hợp với các tiện ích xử lý văn bản như sed, grep. Đây là điểm mạnh quan trọng cho phép tự động hóa quy trình thu thập và phân tích thông tin.

## 4.2 Kết luận chung

Nghiên cứu về công cụ Netcat trên nền tảng Kali Linux trong bối cảnh thu thập thông tin đã khẳng định vai trò nền tảng của công cụ này trong lĩnh vực an toàn thông tin mạng. Netcat không chỉ đơn thuần là một tiện ích dòng lệnh, mà còn là minh chứng điển hình cho sức mạnh của việc tương tác trực tiếp với các giao thức mạng ở tầng thấp.

Về mặt kỹ thuật, giá trị cốt lõi của Netcat nằm ở khả năng thao tác linh hoạt trên các socket TCP và UDP. Trong giai đoạn thu thập thông tin, khả năng này cho phép thực hiện các kỹ thuật định danh dịch vụ (Banner Grabbing) và quét cổng (Port Scanning) với độ chính xác cao và tốc độ nhanh chóng. Sự đơn giản trong kiến trúc của Netcat cho phép nó hoạt động như một thành phần "nguyên thủy" của mạng máy tính, giúp người quản trị và chuyên gia bảo mật hiểu rõ bản chất của quá trình truyền nhận dữ liệu mà không bị che khuất bởi các lớp giao diện phức tạp.

Về khía cạnh an ninh, Netcat mang tính lưỡng dụng rõ rệt. Đối với kiểm thử xâm nhập (Offensive Security), đây là công cụ đắc lực để vẽ lại bản đồ mạng lưới và xác định bề mặt tấn công (Attack Surface). Ngược lại, đối với phòng thủ (Defensive Security), việc phân tích cơ chế hoạt động của Netcat,đặc biệt là đặc điểm truyền dữ liệu dưới dạng văn bản thuần (cleartext),cung cấp cơ sở quan trọng để thiết lập các luật phát hiện xâm nhập (IDS/IPS) và giám sát lưu lượng mạng bất thường.

Tổng hợp lại, mặc dù các công cụ hiện đại với giao diện đồ họa và tính năng tự động hóa ngày càng phát triển, Netcat vẫn giữ vững vị thế là thành phần đa năng trong bộ công cụ Kali Linux. Tuy nhiên, trong bối cảnh các mối đe dọa an ninh mạng ngày càng tinh vi, việc sử dụng Netcat cần được kết hợp với tư duy về bảo mật kênh truyền (như sử dụng Ncat hỗ trợ SSL) và tuân thủ nghiêm ngặt các quy chuẩn đạo đức nghề nghiệp. Việc làm chủ Netcat, do đó, không chỉ là kỹ năng sử dụng công cụ, mà còn là bước đệm thiết yếu để nắm bắt tư duy hệ thống và các nguyên lý vận hành cốt lõi của mạng máy tính.

# Tài liệu tham khảo

**[1]** Fyodor. Nmap Network Scanning: The Official Nmap Project Guide to Network Discovery and Security Scanning. Insecure.Org, 2015.  
**[2]** GNU Netcat Documentation. GNU Netcat Manual.  
<https://nc110.sourceforge.io/>

**[3]** Nmap Project. Ncat — A Modern Netcat Replacement.  
<https://nmap.org/ncat/>

**[4]** OpenBSD Project. OpenBSD Netcat (nc) Manual Page.  
<https://man.openbsd.org/nc>

**[5]** IETF. RFC 793 – Transmission Control Protocol. Internet Engineering Task Force, 1981.  
**[6]** IETF. RFC 768 – User Datagram Protocol. Internet Engineering Task Force, 1980.

**[7]** Offensive Security. Kali Linux Documentation.  
<https://www.kali.org/docs/>

**[8]** Patrick Engebretson. The Basics of Hacking and Penetration Testing, 2nd Edition. Syngress, 2013.