

HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG



BÁO CÁO BÀI THỰC HÀNH 2

Kỹ thuật theo dõi và giám sát an toàn mạng

Lập trình với thư viện pcap

Giảng viên: Ninh Thị Thu Trang

Sinh viên: Hoàng Trung Kiên

Mã sinh viên: B20DCAT098

Hà Nội – 2024

Mục lục

1. Mục đích.	3
2. Yêu cầu đối với sinh viên.	3
3. Nội dung thực hành.....	3
3.1 Dấu vết không rõ ràng.	3
3.2. Thống kê lưu lượng cơ bản.	4
4. Checkwork.	8

1. Mục đích.

- Bài thực hành này giới thiệu về việc lập trình sử dụng thư viện/API pcap.

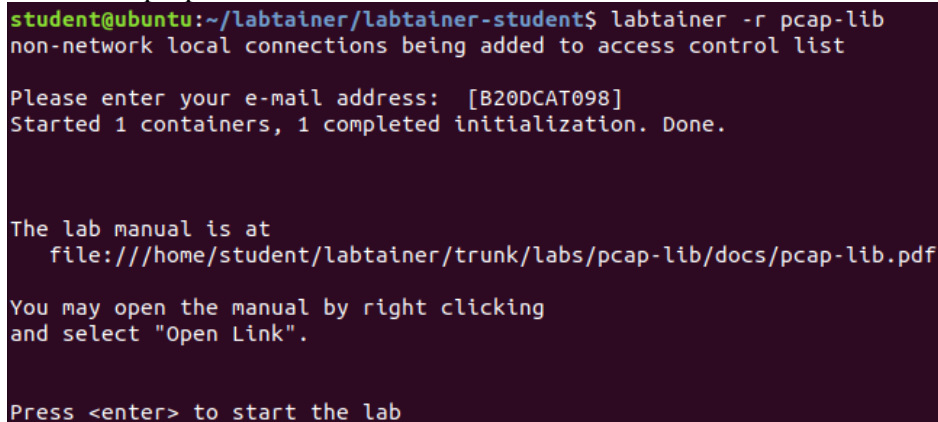
2. Yêu cầu đối với sinh viên.

- Sinh viên có hiểu biết cơ bản về dữ liệu gói tin đầy đủ, định dạng file pcap
- Sinh viên phải có kiến thức cơ bản về dòng lệnh Linux.
- Sinh viên có kỹ năng lập trình cơ bản với C++ hoặc Python.

3. Nội dung thực hành.

Khởi động bài lab: Trên terminal, gõ lệnh:

labtainer -r pcap-lib



```
student@ubuntu:~/labtainer/labtainer-student$ labtainer -r pcap-lib
non-network local connections being added to access control list

Please enter your e-mail address: [B20DCAT098]
Started 1 containers, 1 completed initialization. Done.

The lab manual is at
  file:///home/student/labtainer/trunk/labs/pcap-lib/docs/pcap-lib.pdf

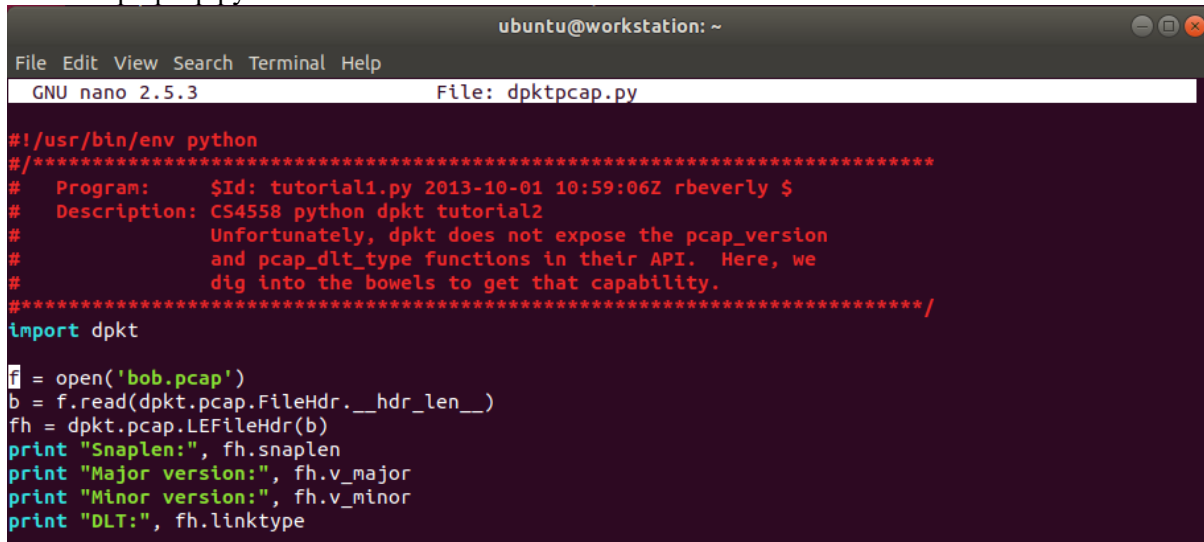
You may open the manual by right clicking
and select "Open Link".

Press <enter> to start the lab
```

3.1 Dấu vết không rõ ràng.

Chúng ta sẽ phân tích một dấu vết gói tin ẩn danh lấy từ một điểm trao đổi Internet. Dấu vết này có trong thư mục gốc của bạn ở định dạng trace2.pcap.

Mở file dpktpcap.py



```
ubuntu@workstation: ~
File Edit View Search Terminal Help
GNU nano 2.5.3 File: dpktpcap.py

#!/usr/bin/env python
# *****
# Program: $Id: tutorial1.py 2013-10-01 10:59:06Z rbeverly $
# Description: CS4558 python dpkt tutorial2
# Unfortunately, dpkt does not expose the pcap_version
# and pcap_dlt_type functions in their API. Here, we
# dig into the bowels to get that capability.
# *****/
import dpkt

f = open('bob.pcap')
b = f.read(dpkt.pcap.FileHdr.__hdr_len__)
fh = dpkt.pcap.LEFileHdr(b)
print "Snaplen:", fh.snaplen
print "Major version:", fh.v_major
print "Minor version:", fh.v_minor
print "DLT:", fh.linktype
```

Thay đổi file bob.pcap thành file trace2.pcap

```
ubuntu@workstation: ~
File Edit View Search Terminal Help
GNU nano 2.5.3 File: dpktpcap.py

#!/usr/bin/env python
#
# *****
# Program: $Id: tutorial1.py 2013-10-01 10:59:06Z rbeverly $
# Description: CS4558 python dpkt tutorial2
#
# Unfortunately, dpkt does not expose the pcap_version
# and pcap_dlt_type functions in their API. Here, we
# dig into the bowels to get that capability.
#
# *****/
import dpkt

f = open('trace2.pcap')
b = f.read(dpkt.pcap.FileHdr.__hdr_len__)
fh = dpkt.pcap.LEFileHdr(b)
print "Snaplen:", fh.snaplen
print "Major version:", fh.v_major
print "Minor version:", fh.v_minor
print "DLT:", fh.linktype
```

=> sau đó chạy file dpktpcap.py

```
ubuntu@workstation: ~
File Edit View Search Terminal Help

ubuntu@workstation:~$ ls
dpktpcap.py  mystuff  pcaplib.cpp  trace2.pcap  tutorial1.py
ubuntu@workstation:~$ nano dpktpcap.py
ubuntu@workstation:~$ python dpktpcap.py
Snaplen: 65535
Major version: 2
Minor version: 4
DLT: 1
ubuntu@workstation:~$
```

- + Loại liên kết trong dấu vết là: Giá trị DLT trong ảnh là 1, biểu thị liên kết Ethernet
- + Snaplen là viết tắt của snapshot length. Nó là một giá trị được sử dụng trong các công cụ ghi lại lưu lượng mạng để xác định số lượng byte được ghi lại từ mỗi gói tin.
- + Sự khác biệt giữa pcap và Pcapng:

Pcap:

- Cũ hơn, phổ biến hơn.
- Ít tính năng hơn, đơn giản hơn.
- Hiệu suất ghi/đọc nhanh hơn.
- Phù hợp cho ghi/phân tích lưu lượng cơ bản.

PcapNg:

- Mới hơn, ít phổ biến hơn.
- Nhiều tính năng hơn, linh hoạt hơn.
- Hiệu suất ghi/đọc chậm hơn.
- Phù hợp cho phân tích chuyên sâu, giải mã, lưu trữ lâu dài.

+ Tỷ lệ giao thức: ko có

3.2. Thống kê lưu lượng cơ bản.

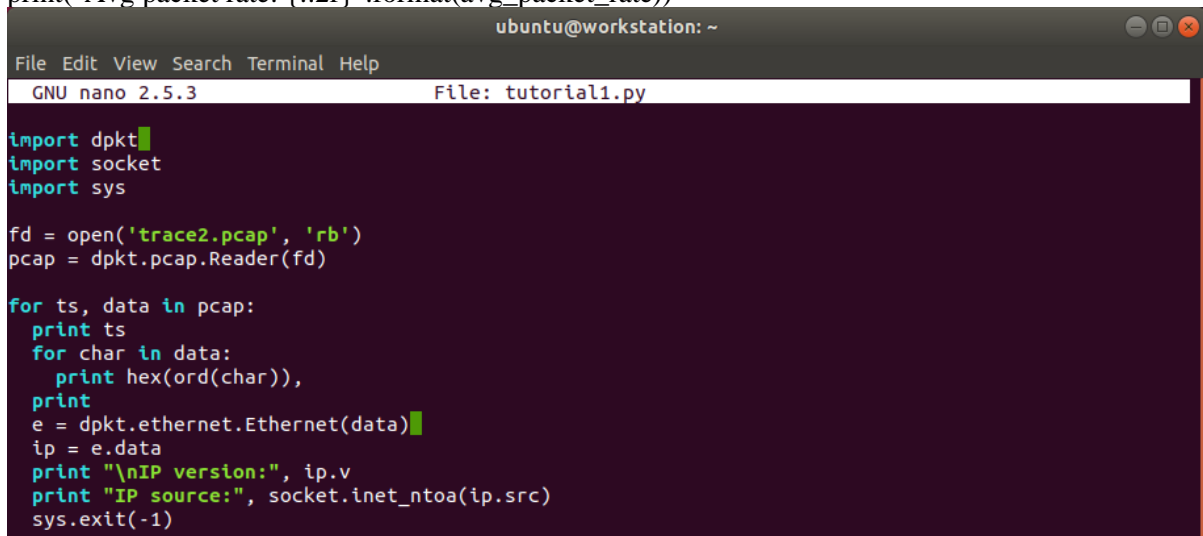
Mở file tutorial1.py và viết đoạn code này vào

```
import dpkt
import socket
fd = open('trace2.pcap', 'rb')
pcap = dpkt.pcap.Reader(fd)
ipv4_count = 0
non_ipv4_count = 0
first_timestamp = None
last_timestamp = None
```

```

packet_count = 0
for ts, data in pcap:
    if first_timestamp is None:
        first_timestamp = ts
    last_timestamp = ts
    eth = dpkt.ethernet.Ethernet(data)
    if isinstance(eth.data, dpkt.ip.IP):
        ipv4_count += 1
    else:
        non_ipv4_count += 1
    packet_count += 1
#Print the results
print("IPv4 count: {}".format(ipv4_count))
print("Non-IPv4 count: {}".format(non_ipv4_count))
#Print the first timestamp with at least two decimal places
print("First timestamp: {:.2f}".format(first_timestamp))
#Calculate and print the average packet rate
time_span = last_timestamp - first_timestamp
avg_packet_rate = packet_count / time_span if time_span > 0 else 0
print("Avg packet rate: {:.2f}".format(avg_packet_rate))

```



```

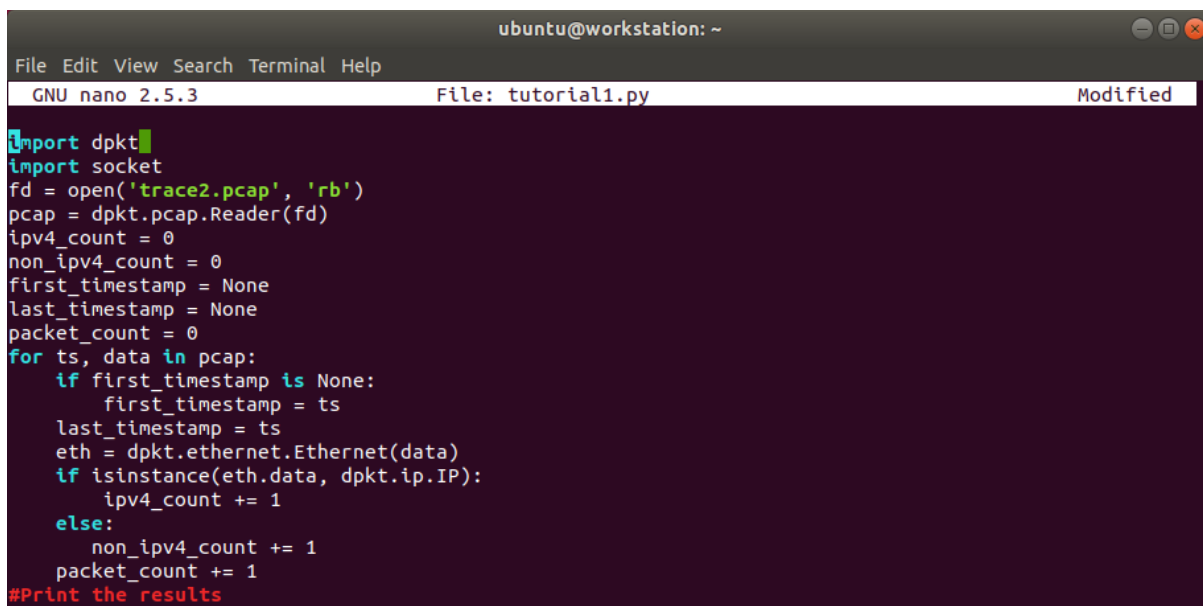
ubuntu@workstation: ~
File Edit View Search Terminal Help
GNU nano 2.5.3 File: tutorial1.py

import dpkt
import socket
import sys

fd = open('trace2.pcap', 'rb')
pcap = dpkt.pcap.Reader(fd)

for ts, data in pcap:
    print ts
    for char in data:
        print hex(ord(char)),
    print
    e = dpkt.ethernet.Ethernet(data)
    ip = e.data
    print "\nIP version:", ip.v
    print "IP source:", socket.inet_ntoa(ip.src)
    sys.exit(-1)

```



```

ubuntu@workstation: ~
File Edit View Search Terminal Help
GNU nano 2.5.3 File: tutorial1.py Modified

import dpkt
import socket
fd = open('trace2.pcap', 'rb')
pcap = dpkt.pcap.Reader(fd)
ipv4_count = 0
non_ipv4_count = 0
first_timestamp = None
last_timestamp = None
packet_count = 0
for ts, data in pcap:
    if first_timestamp is None:
        first_timestamp = ts
    last_timestamp = ts
    eth = dpkt.ethernet.Ethernet(data)
    if isinstance(eth.data, dpkt.ip.IP):
        ipv4_count += 1
    else:
        non_ipv4_count += 1
    packet_count += 1
#Print the results

```

Sau đó chạy file tutorial1.py

```
ubuntu@workstation:~$ python tutorial1.py
ubuntu@workstation: ~
File Edit View Search Terminal Help
ubuntu@workstation:~$ ls
dpktpcap.py  mystuff  pcaplib.cpp  trace2.pcap  tutorial1.py
ubuntu@workstation:~$ nano dpktpcap.py
ubuntu@workstation:~$ python dpktpcap.py
Snaplen: 65535
Major version: 2
Minor version: 4
DLT: 1
ubuntu@workstation:~$ nano tutorial1.py
ubuntu@workstation:~$ python tutorial1.py
IPv4 count: 30611000
Non-IPv4 count: 0
First timestamp: 1474265898.92
Avg packet rate: 708.60
```

- + Dấu vết chứa 30611000 gói tin IPv4
- + Dấu vết chứa 0 gói tin không phải là IPv4
- + Dấu vết có timestamp của gói tin đầu tiên là 1474265898.92
- + Tốc độ trung bình của gói tin trong dấu vết là 708.60.

- Sau tiêu đề pcap cho mỗi gói tin là dữ liệu lưu lượng. Cải thiện chức năng gọi lại của bạn để giải mã các gói tin IP. Chức năng gọi lại của bạn nên thu được địa chỉ IP nguồn và đích, giao thức (ví dụ như TCP, UDP, ICMP, v.v.), cổng vận chuyển nguồn và đích (nếu có), v.v.

- Mở file tutorial1.py và sửa code

```
import dpkt
import socket
```

```
fd = open('trace2.pcap', 'rb')
pcap = dpkt.pcap.Reader(fd)
```

```
unique_source_addresses = set() # Set to store unique source addresses
unique_dest_addresses = set() # Set to store unique destination
```

```
source_bytes = {} # Dictionary to store bytes sent by each source IP
source_packet_counts = {} # Dictionary to store packet counts sent by each source IP
```

```
for ts, data in pcap:
    eth = dpkt.ethernet.Ethernet(data)
    if isinstance(eth.data, dpkt.ip.IP):
        ip = eth.data
        # Convert bytes to string
        if isinstance(ip.src, bytes):
            src_ip = socket.inet_ntoa(ip.src)
            unique_source_addresses.add(src_ip)
            if src_ip in source_bytes:
                source_bytes[src_ip] += len(ip)
            else:
                source_bytes[src_ip] = len(ip)
            if src_ip in source_packet_counts:
                source_packet_counts[src_ip] += 1
            else:
                source_packet_counts[src_ip] = 1
```

```

if isinstance(ip.dst, bytes):
    dst_ip = socket.inet_ntoa(ip.dst)
    unique_dest_addresses.add(dst_ip)

print("Unique sources: {}".format(len(unique_source_addresses)))
print("Unique destinations: {}".format(len(unique_dest_addresses)))

source_with_most_bytes = max(source_bytes, key=source_bytes.get)
source_with_most_packets = max(source_packet_counts, key=source_packet_counts.get)
print("Source with most bytes: {}".format(source_with_most_bytes))
print("Source with most packets: {}".format(source_with_most_packets))

```

```

ubuntu@workstation: ~
File Edit View Search Terminal Help
GNU nano 2.5.3 File: tutorial1.py Modified

import dpkt
import socket

fd = open('trace2.pcap', 'rb')
pcap = dpkt.pcap.Reader(fd)

unique_source_addresses = set() # Set to store unique source addresses
unique_dest_addresses = set() # Set to store unique destination

source_bytes = {} # Dictionary to store bytes sent by each source IP
source_packet_counts = {} # Dictionary to store packet counts sent by each source IP

for ts, data in pcap:
    eth = dpkt.ethernet.Ethernet(data)
    if isinstance(eth.data, dpkt.ip.IP):
        ip = eth.data
        # Convert bytes to string
        if isinstance(ip.src, bytes):
            src_ip = socket.inet_ntoa(ip.src)
            unique_source_addresses.add(src_ip)

```

Sau đó chạy lại file tutorial1.py

```

ubuntu@workstation:~$ python tutorial1.py

ubuntu@workstation:~$ python tutorial1.py
Unique sources: 1018015
Unique destinations: 32771
Source with most bytes: 58.51.150.96
Source with most packets: 58.51.150.96
ubuntu@workstation:~$

```

+ Có bao nhiêu địa chỉ nguồn IPv4 duy nhất trong file dấu vết: 1018015

+ Có bao nhiêu địa chỉ đích IPv4 duy nhất trong dấu vết: 32771

+ IP nguồn nào đã gửi nhiều byte nhất: 58.51.150.96

+ IP nguồn nào đã gửi nhiều gói tin nhất : 58.51.150.96

- Dựa trên phân tích của dấu vết:

+ Liệt kê 3 đặc điểm không bình thường của lưu lượng mạng mà bạn thấy:

+ Tốc độ gói tin cao: Dấu vết cho thấy tốc độ gói tin trung bình là 708,60 gói tin mỗi giây. Đây là tốc độ cao hơn nhiều so với lưu lượng mạng thông thường, có thể là do một cuộc tấn công mạng hoặc hoạt động độc hại khác đang diễn ra.

+ Số lượng gói tin thấp: Mặc dù tốc độ gói tin cao, nhưng số lượng gói tin tổng thể chỉ là 3.061.100. Điều này cho thấy rằng mỗi gói tin có kích thước rất nhỏ, có thể là do một cuộc tấn công quét mạng hoặc hoạt động dò tìm thông tin đang diễn ra.

+ Kích thước gói tin nhỏ: Kích thước gói tin trung bình không được hiển thị trong dấu vết, nhưng có thể là rất nhỏ dựa trên tốc độ gói tin cao và số lượng gói tin thấp. Kích thước gói tin

nhỏ thường được sử dụng trong các cuộc tấn công mạng để tránh bị phát hiện.

+ Dựa trên các đặc điểm không bình thường được xác định, có thể đưa ra một số giải thích hợp lý về lưu lượng mạng mà dấu vết đại diện:

- + Cuộc tấn công từ chối dịch vụ (DoS): Một cuộc tấn công DoS có thể là nguyên nhân khiến tốc độ gói tin cao và số lượng gói tin thấp. Trong một cuộc tấn công DoS, kẻ tấn công sẽ gửi một lượng lớn lưu lượng truy cập đến một máy chủ hoặc mạng, khiến cho máy chủ hoặc mạng quá tải và không thể phục vụ các yêu cầu hợp pháp.
- + Cuộc tấn công quét mạng: Kích thước gói tin nhỏ có thể là dấu hiệu của một cuộc tấn công quét mạng. Trong một cuộc tấn công quét mạng, kẻ tấn công sẽ gửi các gói tin nhỏ đến nhiều địa chỉ IP khác nhau để tìm kiếm các máy tính dễ bị tấn công.
- + Hoạt động dò tìm thông tin: Kích thước gói tin nhỏ cũng có thể được sử dụng trong các hoạt động dò tìm thông tin. Kẻ tấn công có thể sử dụng các gói tin nhỏ để thăm dò mạng và xác định các dịch vụ đang chạy trên các máy tính.

4. Checkwork.

```
Student      | read_trace | ipv4_count | first_timestamp | avg | sources | destinations | src_bytes | src_pkts |
=====|=====|=====|=====|=====|=====|=====|=====|=====|
B28DCAT098   | 3 | Y | Y | Y | Y | Y | Y | Y |

What is automatically assessed for this lab:
  _ipv4_count, _first_timestamp, _avg, _sources, _destinations, _src_bytes, _src_pkts: correct values for the corresponding statistics
  _ipv4_count = *.stdout : FILE REGEX : IPv4 count:.*30611000
  read_trace: Number of times trace2.pcap was opened
  ipv4_count: Count of IPV4 packets
  first_timestamp: First timestamp
  avg: Average packets per second
  sources: Number of unique sources
  destinations: Number of unique destinations
  src_bytes: Source sending most bytes
  src_pkts: Source sending most packets
student@ubuntu:~/labtainer/labtainer-student$
```