

## **Docker-Based SYN Flood Attack Simulation**

Kayvon Karimi

Shirley-Macros School of Engineering (SMSE), University of San Diego

CYBR-508: Secure Network Engineering

Professor Templeton

July 21st, 2025

## Introduction

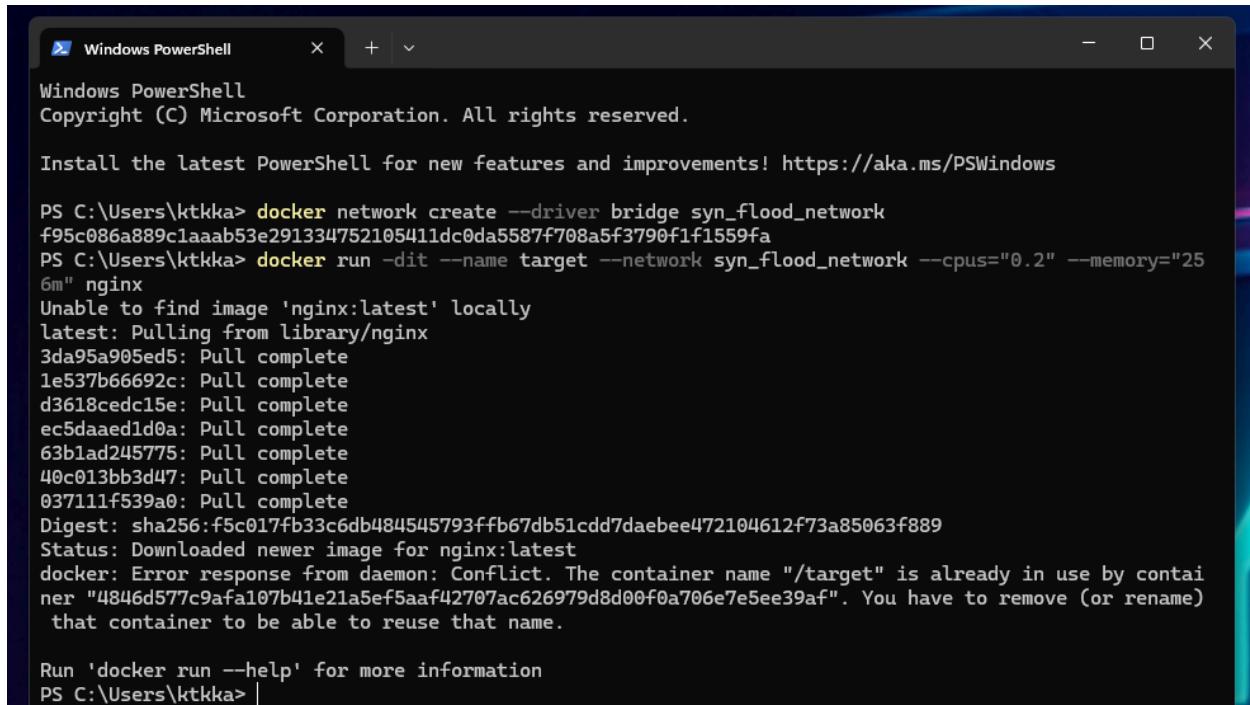
This lab simulated a SYN flood attack using Docker containers to understand Denial of Service (DoS) mechanics. The setup included a target (nginx) and attacker (hping3) on a custom network, monitored with tcpdump and analyzed in Wireshark.

### Step 1: Set up Docker Containers

- **Created network:**

Open Docker on home device and open Powershell.

In Powershell, create a network with '*docker network create --driver bridge syn\_flood\_network*'



```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ktkka> docker network create --driver bridge syn_flood_network
f95c086a889c1aaab53e291334752105411dc0da5587f708a5f3790f1f1559fa
PS C:\Users\ktkka> docker run -dit --name target --network syn_flood_network --cpus="0.2" --memory="25
6m" nginx
Unable to find image 'nginx:latest' locally
latest: Pulling from library/nginx
3da95a905ed5: Pull complete
1e537b66692c: Pull complete
d3618cedc15e: Pull complete
ec5daaaed1d0a: Pull complete
63b1ad245775: Pull complete
40c013bb3d47: Pull complete
037111f539a0: Pull complete
Digest: sha256:f5c017fb33c6db484545793ffb67db51cdd7daebee472104612f73a85063f889
Status: Downloaded newer image for nginx:latest
docker: Error response from daemon: Conflict. The container name "/target" is already in use by container "4846d577c9afa107b41e21a5ef5aaaf42707ac626979d8d00f0a706e7e5ee39af". You have to remove (or rename)
that container to be able to reuse that name.

Run 'docker run --help' for more information
PS C:\Users\ktkka> |
```

- **Configured and configured target:**

- *'docker run -dit --name target --network syn\_flood\_network --cpus="0.2" --memory="256m" nginx'*

- apt update && apt install iproute2 -y
- mkdir -p /mnt/data
- apt update && apt install tcpdump -y

```
Run 'docker run --help' for more information
PS C:\Users\ktkka> docker exec -it target bash
root@de08ccb097b2:/# apt update && apt install iproute2 -y
Get:1 http://deb.debian.org/debian/buster/main iproute2 amd64 4.14.19-1+b1 [151 kB]
```

```
root@de08ccb097b2:/# mkdir -p /mnt/data
root@de08ccb097b2:/# apt update && apt install tcpdump -y
```

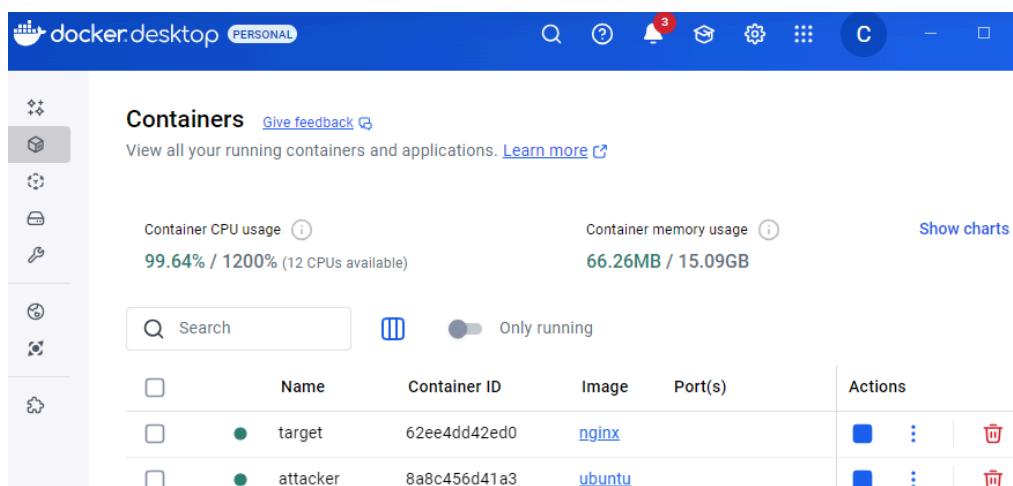
- Created attacker:

*'docker run -it --name attacker --network syn\_flood\_network ubuntu /bin/bash'*

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Install the latest PowerShell for new features and improvements! https://aka.ms/PSWindows

PS C:\Users\ktkka> docker start attacker
attacker
PS C:\Users\ktkka> docker run -it --name attacker --network syn_flood_network ubuntu /bin/bash
root@87fe29ff1e9d:/# apt update && apt install hping3 -y
```



Docker Application on desktop shows target and attacker containers running.

## Step 2: Launch the Attack

I ran '*hping3 --flood --syn --destport 80*' target for 30-60 seconds and stopped with *ctrl+c*.

```
root@87fe29ff1e9d:/# hping3 --flood --syn --destport 80 target
HPING target (eth0 172.20.0.2): S set, 40 headers + 0 data bytes
hping in flood mode, no replies will be shown
```

## Step 3: Monitor with tcpdump

Used the command '*tcpdump -i eth0 -c 980 'tcp[tcpflags] == tcp-syn' and not port 22 -w /mnt/data/syn\_flood\_capture.pcap*' during attack.

```
PS C:\Users\ktkka> docker exec -it target bash -c "tcpdump -i eth0 'tcp[tcpflags] == tcp-syn' and not port 22 -w /mnt/data/syn_flood_capture.pcap -c 980"
tcpdump: listening on eth0, link-type EN10MB (Ethernet), snapshot length 262144 bytes
980 packets captured
1855 packets received by filter
0 packets dropped by kernel
```

To monitor the SYN flood attack, I re-entered the target container and ran *tcpdump* to capture SYN packets. The command used was *tcpdump -i eth0 'tcp[tcpflags] == tcp-syn' and not port 22 -w /mnt/data/syn\_flood\_capture.pcap*, which listened on the *eth0* interface and saved the capture to a PCAP file. During the attack, I also monitored bandwidth usage with *iftop* in a separate window, which showed increased traffic from the attacker IP (151.101.202.132) to the target.

```

Windows PowerShell

          191Mb           381Mb           572Mb           763Mb           954Mb
6c820f3ae6fb          => attacker.syn_flood_network
                         <= 
6c820f3ae6fb          => 151.101.202.132
                         <= 

TX:          cumb: 47.6MB   peak: 18.3Mb      rates: 5.37Mb 4.16Mb 1.82Mb
RX:          86.9MB      18.8Mb      9.76Mb 7.45Mb 1.86Mb
TOTAL:        134MB       29.1Mb      15.1Mb 11.6Mb 2.96Mb

```

The tcpdump capture confirmed 980 packets, as limited by the -c 980 option in a 30 sec span.

```

Windows PowerShell

          2.0%] 3[|          0.7%] 6[          0.0%] 9[|          2.0%]
          2.0%] 4[|          1.3%] 7[|          0.7%] 10[|          1.3%
          2.0%] 5[|          1.3%] 8[|          2.6%] 11[|          2.0%
Mem[|||||||]          908M/15.5G] Tasks: 16, 3 thr, 0 kthr; 1 running
Swp[                  0K/4.00G] Load average: 0.05 0.04 0.09
                                         Uptime: 00:59:39

Main I/O CPU% MEM% TIME+ Command
PID USER PRI NI VIRT RES SHR S
 1 root  20  0 11468 7296 6144 S 0.0 0.0 0:00.02 nginx: master process nginx -g daemon off;
29 nginx 20  0 11932 2924 1280 S 0.0 0.0 0:00.00 nginx: worker process
30 nginx 20  0 11932 2924 1280 S 0.0 0.0 0:00.00 nginx: worker process
31 nginx 20  0 11932 2928 1280 S 0.0 0.0 0:00.00 nginx: worker process
32 nginx 20  0 11932 2928 1280 S 0.0 0.0 0:00.00 nginx: worker process
33 nginx 20  0 11932 2928 1280 S 0.0 0.0 0:00.00 nginx: worker process
34 nginx 20  0 11932 2928 1280 S 0.0 0.0 0:00.00 nginx: worker process
35 nginx 20  0 11932 2928 1280 S 0.0 0.0 0:00.00 nginx: worker process
36 nginx 20  0 11932 2928 1280 S 0.0 0.0 0:00.00 nginx: worker process
37 nginx 20  0 11932 2928 1280 S 0.0 0.0 0:00.00 nginx: worker process
38 nginx 20  0 11932 2928 1280 S 0.0 0.0 0:00.00 nginx: worker process
39 nginx 20  0 11932 2928 1280 S 0.0 0.0 0:00.00 nginx: worker process
40 nginx 20  0 11932 2928 1280 S 0.0 0.0 0:00.00 nginx: worker process
41 root  20  0 4192 3072 2688 S 0.0 0.0 0:00.01 bash
567 root  20  0 225M 36352 5376 S 0.0 0.2 0:08.59 iftop -i eth0
713 root  20  0 225M 36352 5376 S 0.0 0.2 0:00.00 iftop -i eth0
714 root  20  0 225M 36352 5376 S 0.0 0.2 0:00.00 iftop -i eth0
715 root  20  0 225M 36352 5376 S 0.0 0.2 0:06.83 iftop -i eth0
866 root  20  0 4660 3584 2944 R 0.0 0.0 0:00.82 htop

```

#### Step 4: Copied file and Opened Wireshark

- Used the ‘`docker cp target:/mnt/data/syn_flood_capture.pcap`.’ command to copy the file.
- Opened Wireshark, filtered for SYN packets

```
PS C:\Users\ktkka> docker cp target:/mnt/data/syn_flood_capture.pcap .
Successfully copied 70.7kB to C:\Users\ktkka\.
```

- Observed a high volume of SYN packets without ACKs, confirming flood

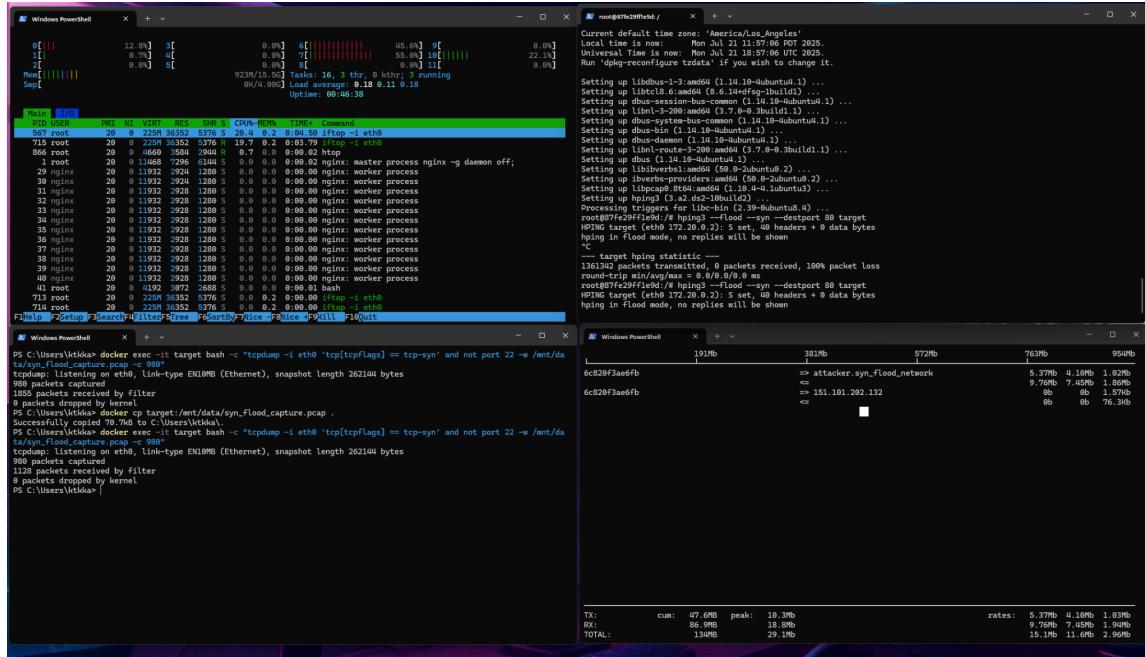
1 0.000000	172.20.0.2	52104 2a:2d:7d.. 151.101.202.132	80 52:3b:c4..	TCP
2 6.837862	172.20.0.2	38946 2a:2d:7d.. 151.101.202.132	80 52:3b:c4..	TCP
3 18.770833	172.20.0.3	1899 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
4 18.770898	172.20.0.3	1900 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
5 18.770912	172.20.0.3	1901 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
6 18.770922	172.20.0.3	1902 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
7 18.770931	172.20.0.3	1903 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
8 18.770941	172.20.0.3	1904 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
9 18.770950	172.20.0.3	1905 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
10 18.770959	172.20.0.3	1906 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
11 18.770968	172.20.0.3	1907 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
12 18.770976	172.20.0.3	1908 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
13 18.770985	172.20.0.3	1909 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
14 18.770993	172.20.0.3	1910 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
15 18.771004	172.20.0.3	1911 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
16 18.771013	172.20.0.3	1912 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP
17 18.771022	172.20.0.3	1913 de:88:e1.. 172.20.0.2	80 2a:2d:7d..	TCP

[Stream index: 0]  
[Stream Packet Number: 1]  
> [Conversation completeness: Incomplete, SYN\_SENT (1)]  
[TCP Segment Len: 0]  
Sequence Number: 0 (relative sequence number)  
Sequence Number (raw): 2752361498  
[Next Sequence Number: 1 (relative sequence number)]  
Acknowledgment Number: 0  
Acknowledgment number (raw): 0  
1010 .... = Header Length: 40 bytes (10)  
> Flags: 0x002 (SYN)  
Window: 64240  
[Calculated window size: 64240]  
Checksum: 0xe0ef [unverified]  
[Checksum Status: Unverified]  
Urgent Pointer: 0  
> Options: (20 bytes), Maximum segment size, SACK permitted, Timestamps, No-Operation (NOP), Window scale  
> [Timestamps]

▼ Transmission Control Protocol, Src Port: 2256, Dst Port: 80, Seq: 0, Len: 0

Source Port: 2256  
Destination Port: 80  
[Stream index: 359]  
[Stream Packet Number: 1]  
> [Conversation completeness: Incomplete, SYN\_SENT (1)]  
[TCP Segment Len: 0]  
Sequence Number: 0 (relative sequence number)  
Sequence Number (raw): 522737853  
[Next Sequence Number: 1 (relative sequence number)]  
▼ Acknowledgment Number: 404518908  
> [Expert Info (Note/Protocol): The acknowledgment number field is nonzero while the ACK flag is not set]  
Acknowledgment number (raw): 404518908  
0101 .... = Header Length: 20 bytes (5)  
> Flags: 0x002 (SYN)  
Window: 512  
[Calculated window size: 512]  
Checksum: 0x4497 [unverified]  
[Checksum Status: Unverified]

Screenshot overview of 4 windows - target, attacker, tcpdump, and ftop.



## Step 5: Clean-Up

Ran '`docker stop target attacker`, `docker rm target attacker`, `docker network rm syn_flood_network`' to clean and remove the containers.

```
PS C:\Users\ktkka> docker container stop target attacker
target
attacker
PS C:\Users\ktkka> docker container rm target attacker
target
attacker
PS C:\Users\ktkka> docker network rm syn_flood_network
syn_flood_network
```

## Analysis

This lab walked us through a hands-on experience in simulating a SYN flood attack within a controlled Docker environment. It enhanced my understanding of Denial of Service mechanics

and network security principles. The successful setup of four terminals, target, attacker, tcpdump, and iftop, allowed for real time monitoring of the attack's impact. The wireshark analysis of the syn\_flood\_capture.pcap file revealed a high volume of SYN packets without corresponding ACKs, which confirms the flood's effectiveness in overwhelming the target's limited resources. The iftop window highlights bandwidth spikes, reinforcing the attack's network load. Lastly, the tcpdump window captured the packet flow, limited to 980 packets for manageable analysis. The impact of the attack was visible through increased CPU usage and high bandwidth consumption and rapid growth of captured SYN packets, which are clear indicators of strained systems.

SYN flood attacks are especially dangerous because of their simplicity and low barrier to execution. In practical terms, this type of attack poses serious implications for online services, particularly for websites, e-commerce platforms, or critical infrastructure relying on uninterrupted access. A successful SYN flood could lead to financial losses, reputational damage, and compromised user trust.