Firewalls

Network Traffic

Five Key Elements (The Five-Tuple)

Source IP (SIP) – the ip address of the device originating the traffic.

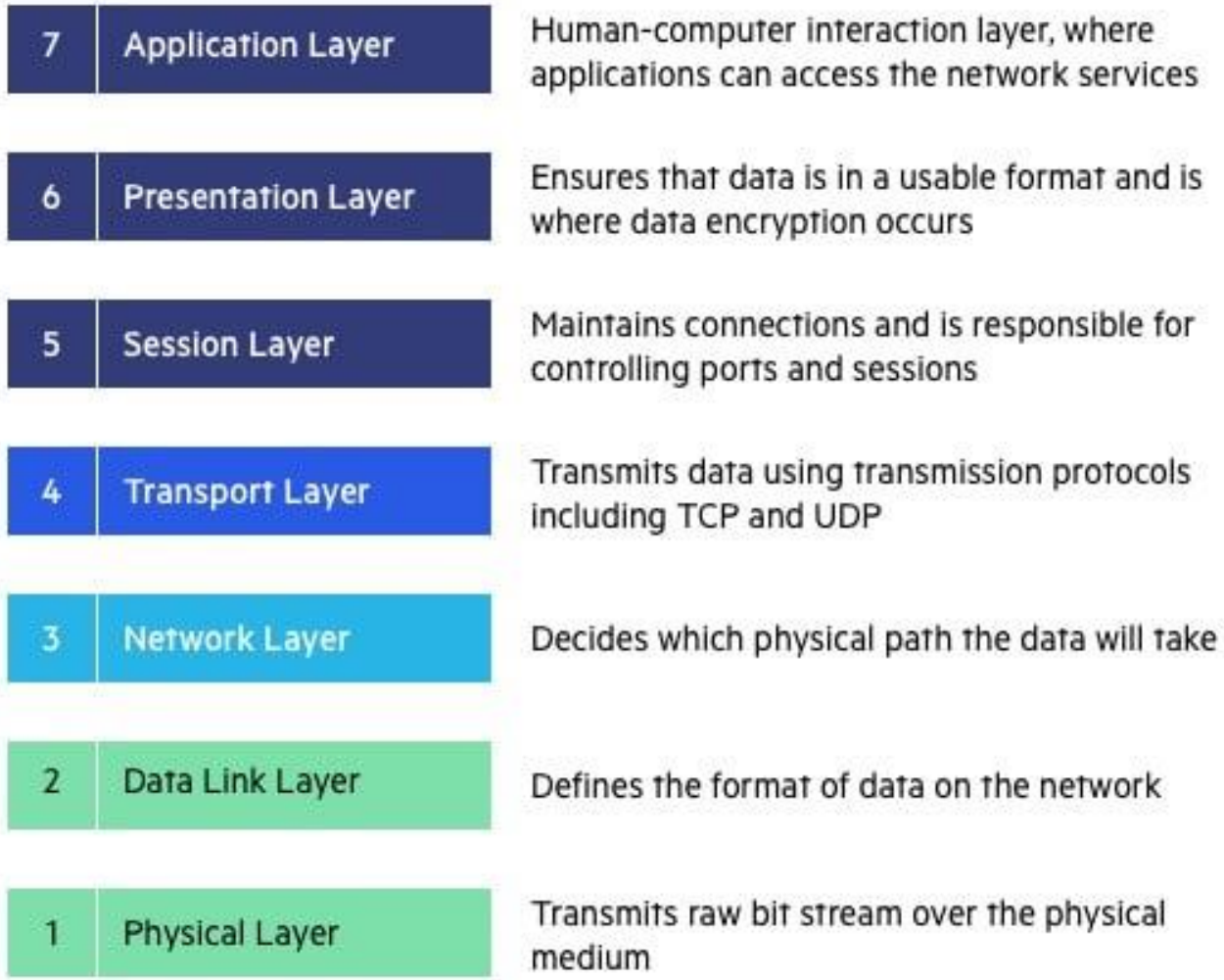
Destination IP (DIP) – the ipaddress of the device that the traffic is destined for.

Source Port (Sport) – The port number on the device at the SIP on which it is waiting for an answer (the answer port). This is usually either an ephemeral port or, for some services, the same as the destination port.

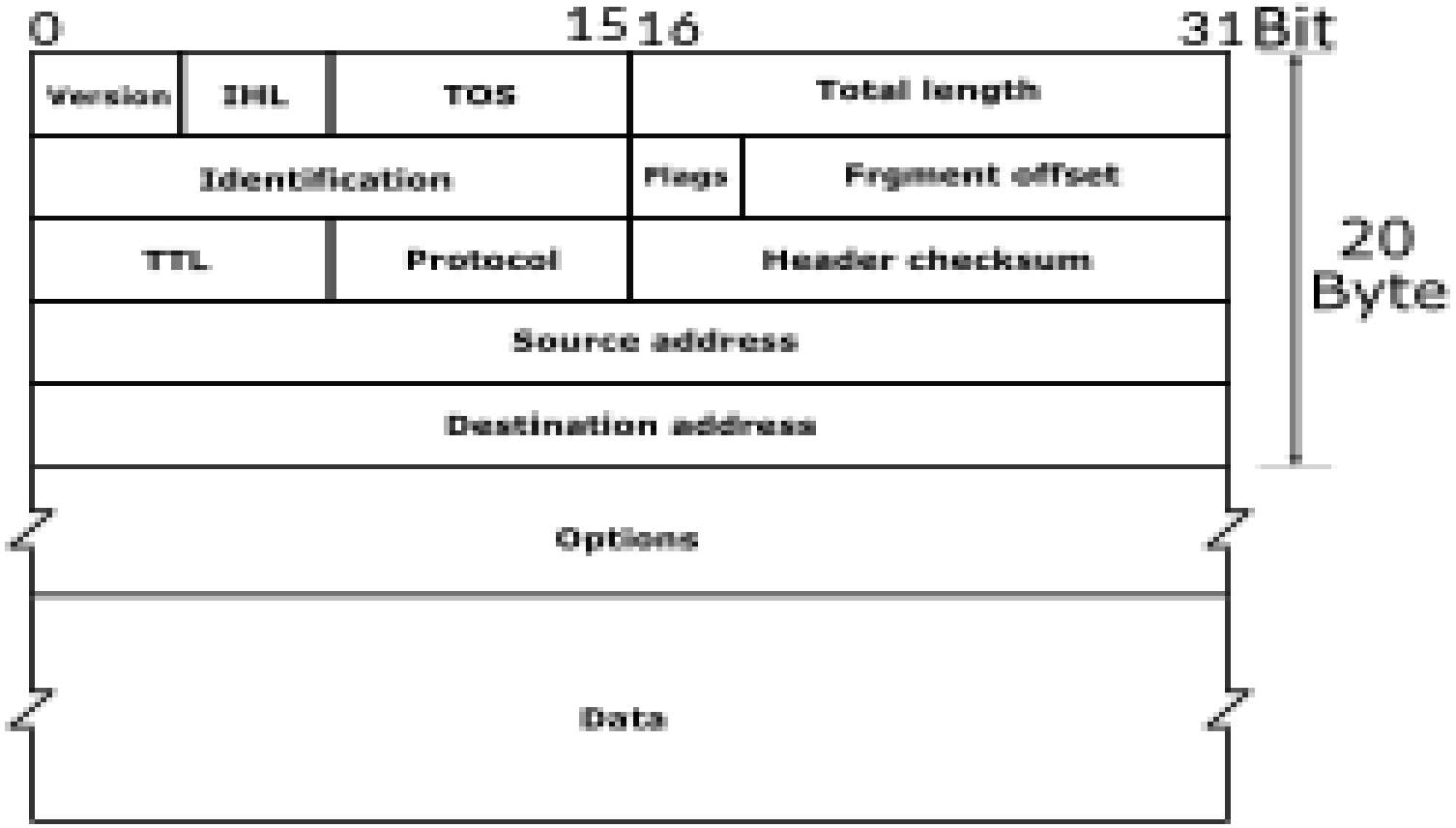
Destination Port (Dport)– The Port number on the DIP to which the traffic is sent.

Protocol (Proto)– The IP protocol number which describes the protocol the traffic is using

OSI Network Stack Model



The IP Header



* https://en.wikipedia.org/wiki/IPv4

Example Five Tuple

SIP: 10.0.0.5

Sport: 50000

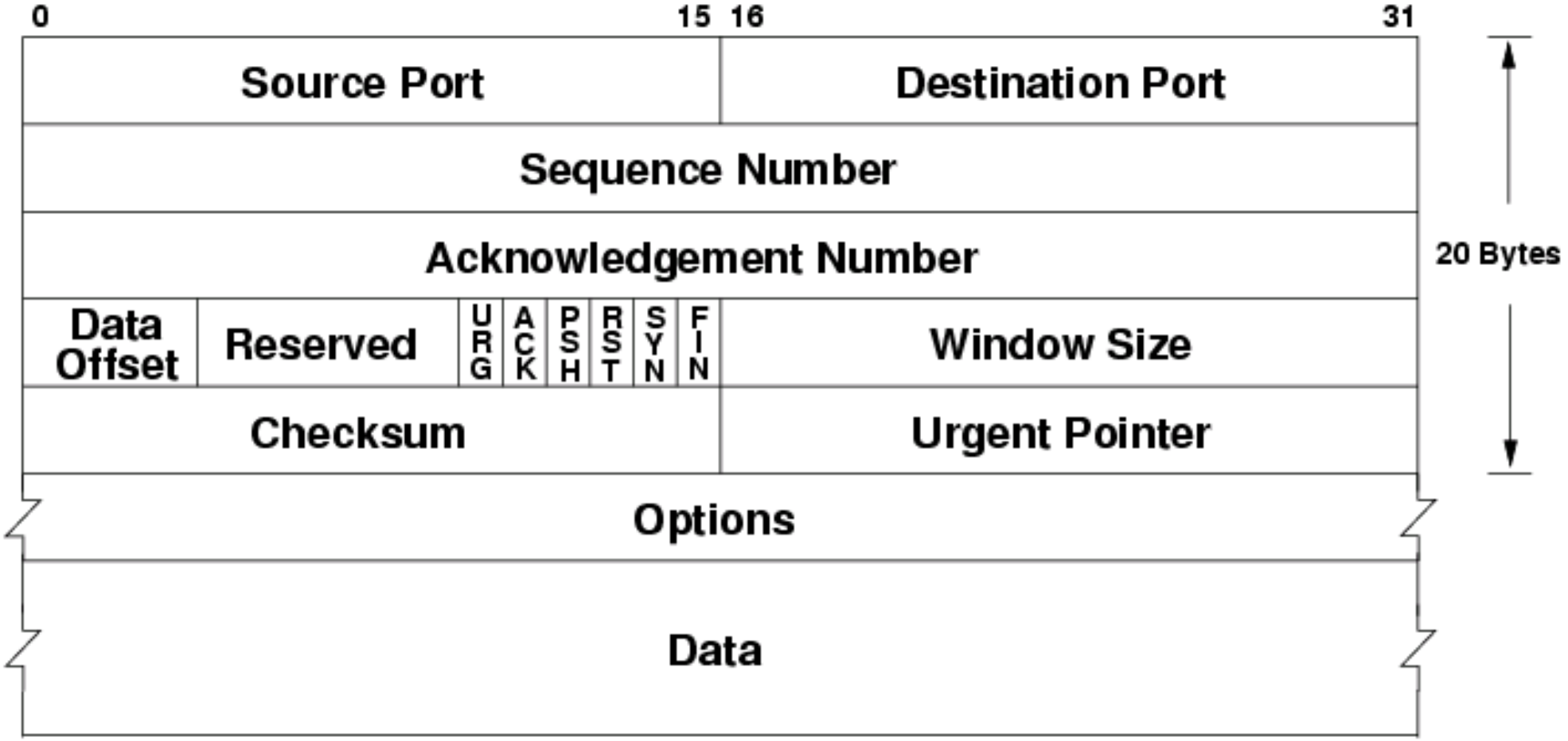
DIP: 10.0.0.8

Dport: 80

Proto: 6

*Protocol #6 refers to TCP

The TCP Header



* https://commons.wikimedia.org/wiki/File:TCP_header.png

Example Five Tuple

SIP: 10.0.0.5

Sport: 50000

DIP: 10.0.0.8

Dport: 2017

Proto: 17

*Protocol #17 refers to TCP

Example Five Tuple

SIP: 10.0.0.5

Sport: 2048

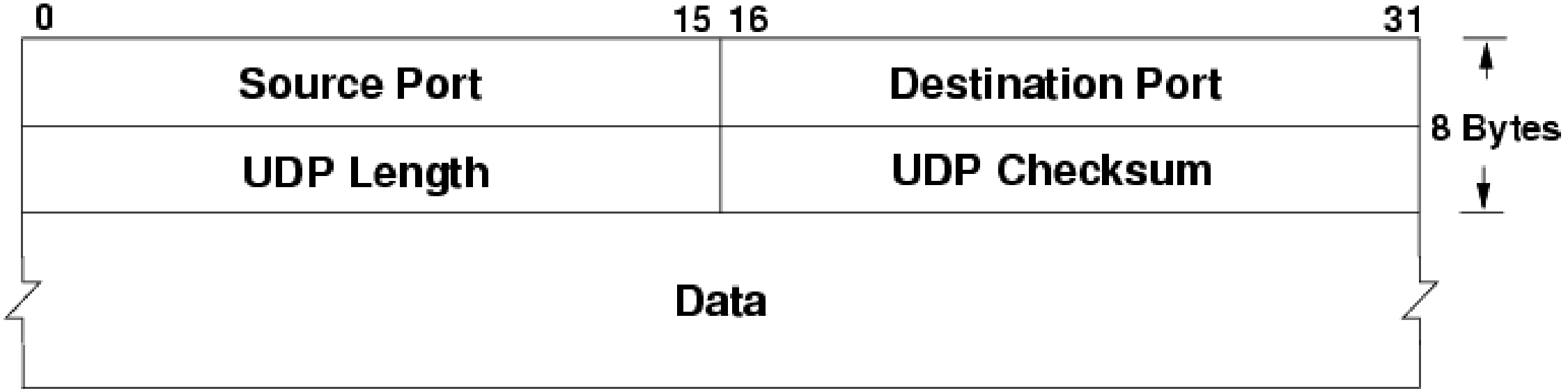
DIP: 10.0.0.8

Dport: 2017

Proto: 17

*Protocol #17 refers to TCP

The UDP Header



*https://en.wikipedia.org/wiki/Internet_Control_Message_Protocol

A note on ICMP ports

ICMP (protocol #1) does not uses Port Numbers

Instead, the Port Number fields of the IP Header are used to encode the ICMP Type & Code.

Type is encoded as an 8-bit number ranging from 0-255

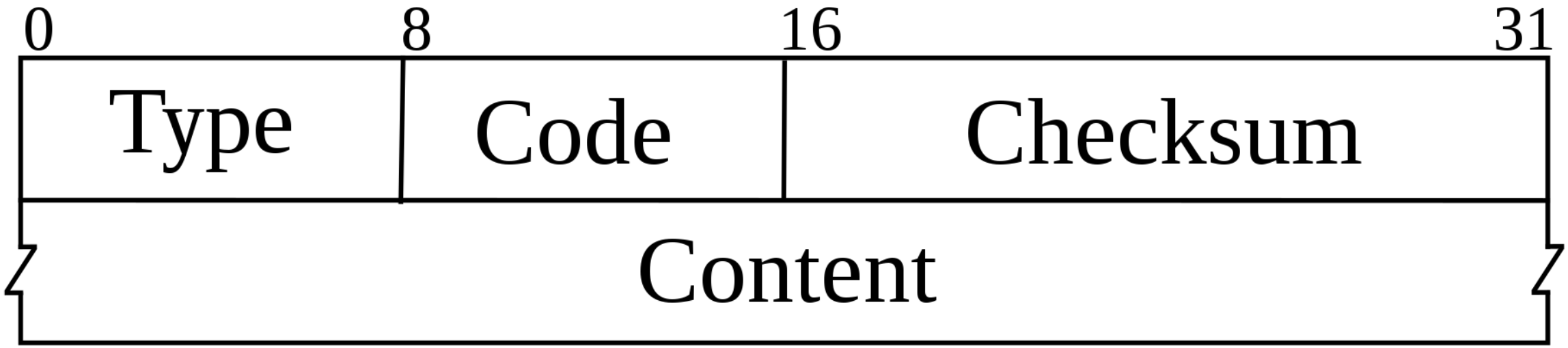
Code is also encoded as an 8-bit number ranging from 0-255

Together they occupy 16 bits in the IP Header

Example: Type 0 Code 0 = Echo Reply (Ping)

Type 3 Code 1 = Host Unreachable

The ICMP Header



*https://commons.wikimedia.org/wiki/File:UDP_header.png

A note on ICMP ports

On some Traffic Collection systems these are grouped together to form a 16 bit “port number” which you have to decode

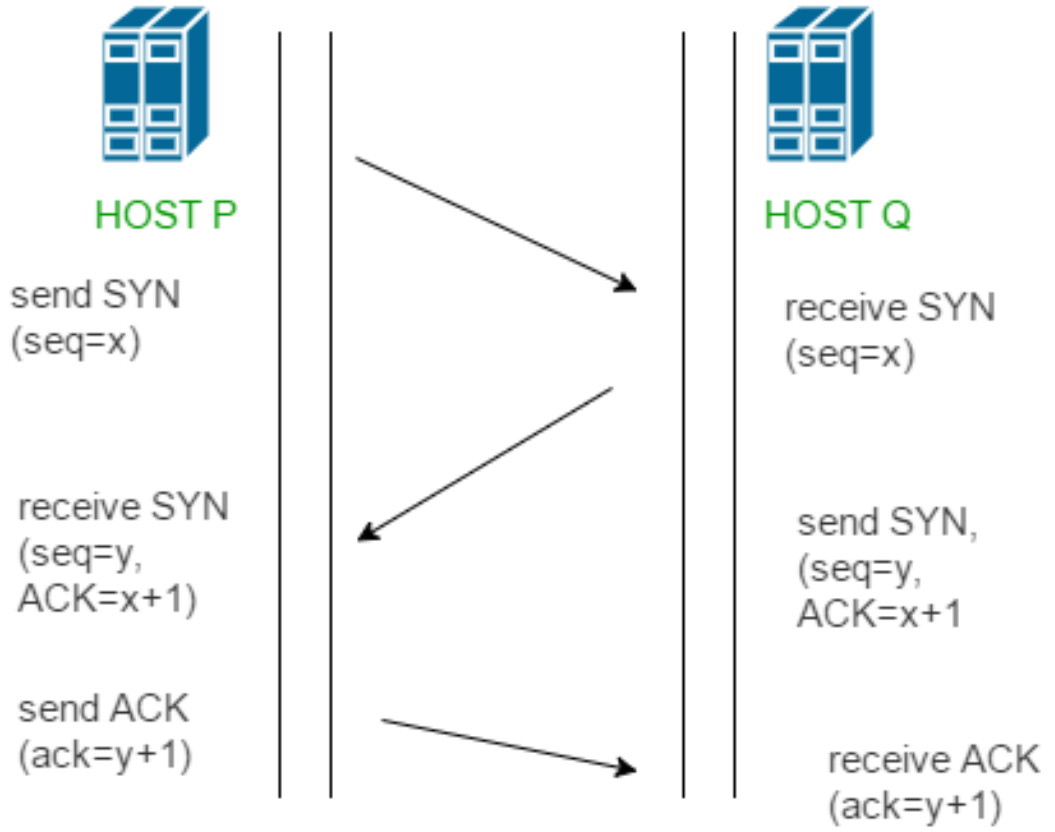
Example: Type 3 Code 1 = Host Unreachable

Type 3 Code 1

0000 0011 0000 0001

Displayed as 0000001100000001 = Port Number 769

The TCP 3-way Handshake



*https://www.geeksforgeeks.org/tcp-3-way-handshake-process/

Types of Firewalls

- Host based vs Network Based
- Packet Filters (1987)
- Stateful (1990)
- Application (1993)
- NextGen (2012)

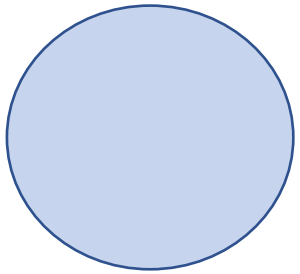
Types of Firewalls

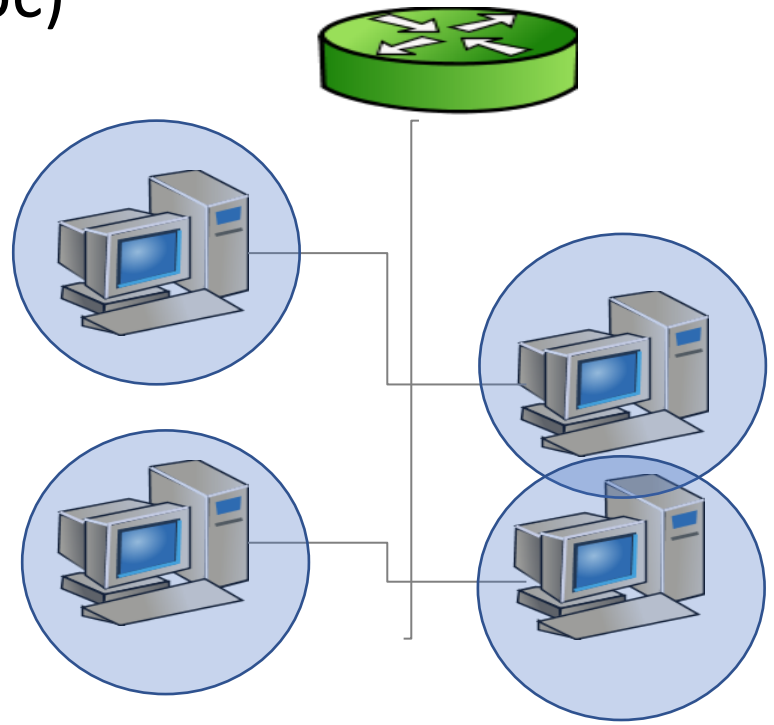
Host Based

Host Based Firewalls exist as software on the device that is communicating with the network (i.e. your pc)

e.g. iptables in linux

Location of the firewall
Is shown as a blue circle





Types of Firewalls

Host Based

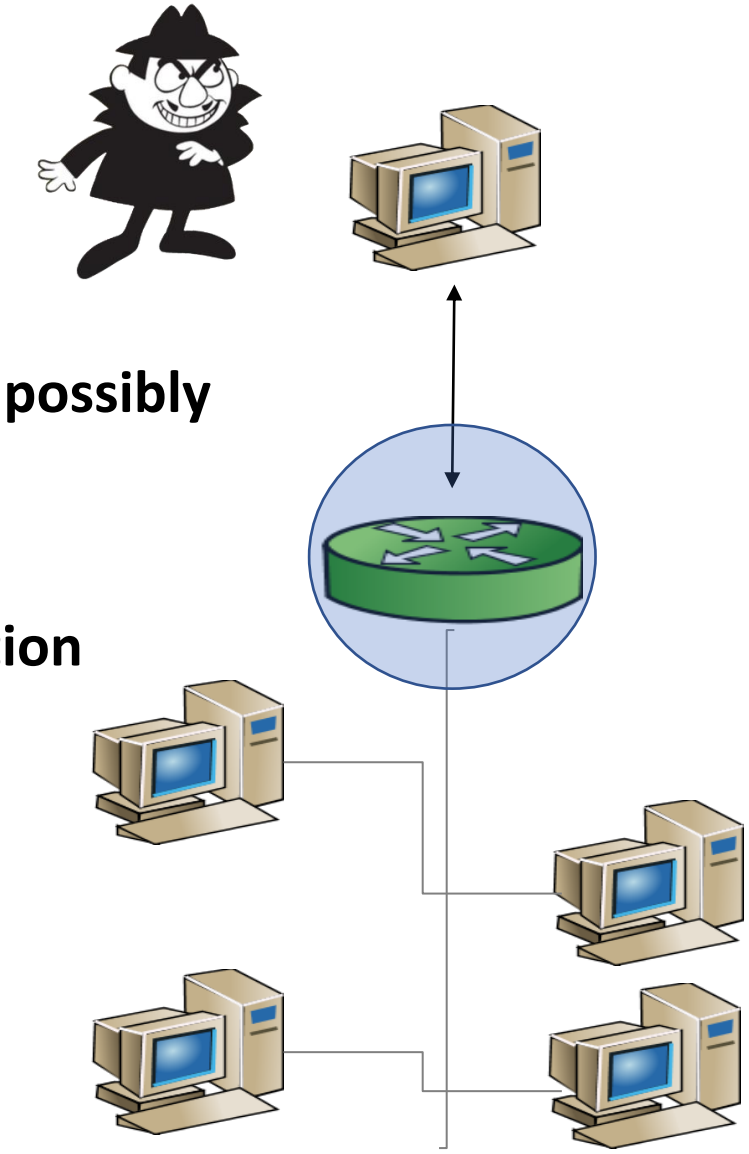
Pluses

- **User configurable**
- **Can be tailored to the operating system**
- **Last line of defence before your endpoint protection system**
- **May have to stop outbound traffic from the host.**
- **Can include correlation using PID numbers for increased filtering on applications.**

Minuses

- **You don’t necessarily want the user in control.**
- **May not be as effective as more specialized, expensive solutions.**
- **Not the best for standardization and scaling.**
- **May have to stop outbound traffic from the host.**

Types of Firewalls



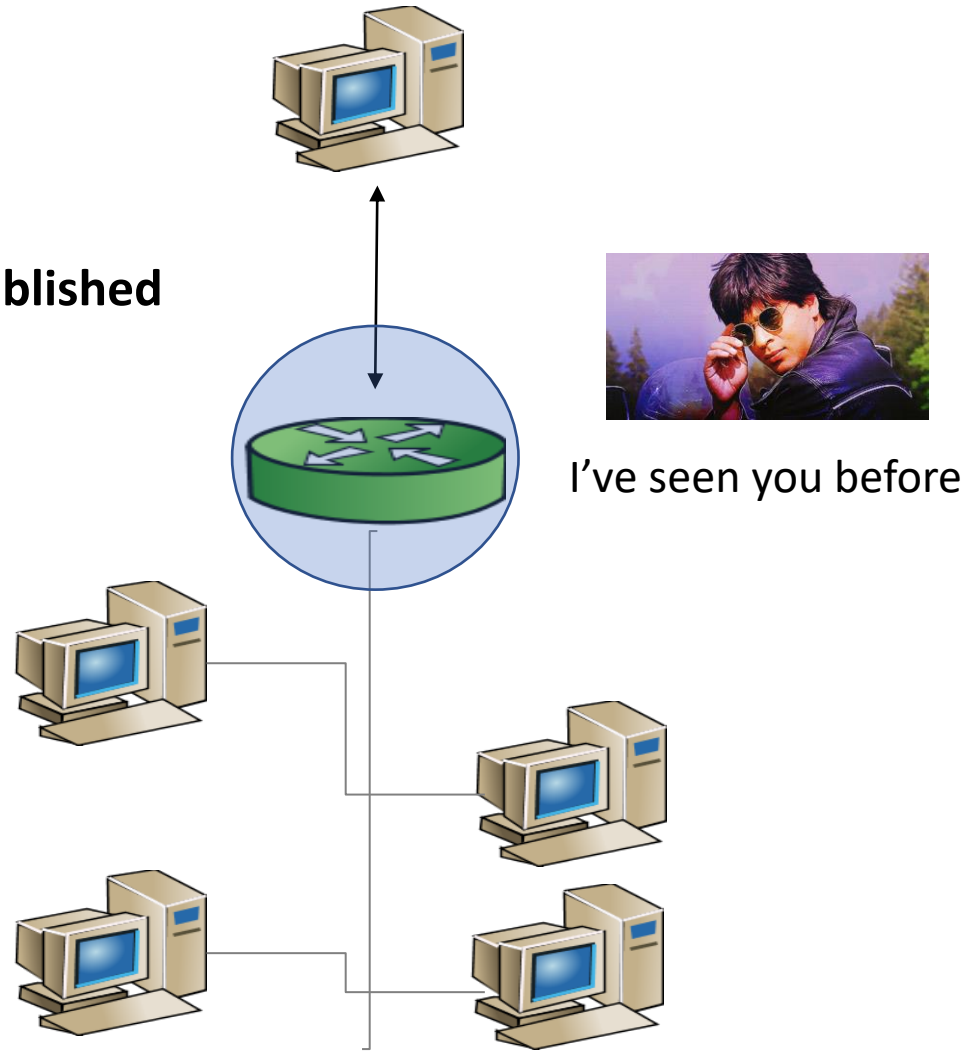
Network Based

- **Keeps unwelcome traffic from reaching and possibly enumerating your hosts.**
- **Often contains specialized solutions in addition to basic packet filtering.**
- **Enables centralized management.**

Types of Firewalls

Stateful

- **Remembers a conversation that has already been established and approved.**
- **Reduces overhead on the firewall.**



I’ve seen you before

Types of Firewalls

Application

- **Can Filter based on Application Identification.**
- **Interprets protocols to discover what application created the traffic.**
- **Can discover applications using non-standard ports.**

Types of Firewalls

NextGen Firewalls

Combine several previously separate functions into a single firewall.

- Packet Filtering
- IPS
- Deep Packet Inspection
- Identity Management
- Application Filtering

Linux Firewalls

IP Firewall (ipfw)

- The first linux firewall. (packet filter)
- Kernel module
- The first firewall chosen for MAC OSx
- Replaced by **ipchains**

Ipchains

A system for controlling the packet filtering functions of the linux kernel.

Expanded the capabilities of ipfw (i.e. able to recognize more protocols).

Replaced by **iptables**

Linux Firewalls

iptables

Acts as a user interface to **netfilter.**

netfilter is a kernel level frame that provides hooks in various areas of the network stack to control packets.

netfilter is made up of **tables** (the **filter** table is the one used for firewalling).

Each table is made up of **chains**.

Each chain is made up of **rules.**

Also available are *ip6tables, arptables and ebtables.*

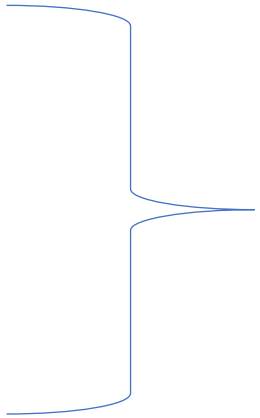
Linux Firewalls

Netfilter

Table

Chain

Rules



iptables

Chains of Rules are written and modified using **iptables**

Linux Firewalls

iptables

- Each rule defines what to do with a packet if it matches the rule.
- Packets that are matched are given a **target**.
- Targets can be another **chain**, or a **special value**.
  - **Special Values** Include:
    - **ACCEPT** packet is allowed to proceed
    - **DROP** packet is not allowed to proceed (is quiet)
    - **RETURN** skip this chain and go back to the next rule in the previous chain.
    - **REJECT** packet is not allowed to proceed and a “port unreachable” is sent.

Linux Firewalls

iptables

Command sequence is:

sudo iptables -<A/I/D> -<CHAIN> -i <INTERFACE> -p <PROTOCOL> -s <SOURCE> -d <DESTINATION> --dport <PORT NUMBER> -j <JUMP TARGET>

A = append

I = insert

D = drop

Linux Firewalls

iptables

Command sequence is:

sudo iptables -<A/I/D> -<CHAIN> -i <INTERFACE> -p <PROTOCOL> -s <SOURCE> -d <DESTINATION> --dport <PORT NUMBER> -j <JUMP TARGET>

one letter -> one dash

more than one -> two dashes

Chains are:

INPUT – incoming packets

OUTPUT – outgoing packets

FORWARD – we won't be using this chain for now.

Linux Firewalls

iptables

Command sequence is:

sudo iptables -<A/I/D> -<CHAIN> -i <INTERFACE> -p <PROTOCOL> -s <SOURCE> -d <DESTINATION> --dport <PORT NUMBER> -j <JUMP TARGET>

JUMP TARGETS include the following *special values*:

- **ACCEPT** packet is allowed to proceed
- **DROP** packet is not allowed to proceed (is quiet)
- **RETURN** skip this chain and go back to the next rule in the previous chain.
- **REJECT** packet is not allowed to proceed and a “port unreachable” is sent.

Linux Firewalls

iptables

Examples:

To drop inbound ftp

#define a variable for the server ip

SERVER_IP=”192.168.0.3”

sudo iptables -I INPUT -i eth0 -p tcp -s 0/0 -d $SERVER_IP --dport 21 -j DROP

To allow inbound SSH

#define a variable for the network

NETWORK = “192.168.0.0/24”

sudo iptables -I INPUT -i eth0 -p tcp -s $NETWORK -d $SERVER_IP --dport 22 -j ACCEPT

Useful Commands

To list the (1) default policies and (2) current status for iptables respectively:
1 just shows default policies
sudo iptables –L

2 to list the default policies and stats v=verbose
sudo iptables –L –v

To set the policies on the CHAINS:
sudo iptables –policy INPUT DROP
sudo iptables –policy OUTPUT ACCEPT
sudo iptables –policy FORWARD DROP

Useful Commands

To find the iptables binary in your linux distro

whereis iptables

If you need to install iptables

sudo apt-get install iptables

To save your new iptables rules to disk

sudo iptables-save

To save rules to a new file

sudo iptables-save > *<u>/path/filename</u>*

To restore rules from a file

sudo iptables-restore < *<u>/path/filename</u>*

Useful Commands

To flush all your existing iptables rules and start over **<BE CAREFUL>**

sudo iptables –F

To flush only the rules from the INPUT CHAIN

sudo iptables –F INPUT

Useful Commands

To add “statefullness” to outbound connections

sudu iptables –A OUTPUT –m conntrack --ctstate NEW, ESTABLISHED –j ACCEPT

To add statefullness to inbound connections

sudu iptables –A INPUT –m conntrack --ctstate ESTABLISHED, RELATED –j ACCEPT

Useful Commands

To see line numbers of rules

sudo iptables -L --line-numbers

To delete a specific rule

sudo iptables -D <CHAIN> <line number>

sudo iptables -D INPUT 3