Firewall

Author: Przemysław Piotrowski
Email: przemyslav.piotrowski@gmail.com

Firewall

What is it?

Firewall

What is it?

Firewall is a hardware or software that prevents unauthorized access to or from a network

Firewall

What is it?

Firewall is a hardware or software that prevents unauthorized access to or from a network

----- Too vague and unclear

Firewall

What is it?

Firewall is a hardware or software that prevents unauthorized access to or from a network

----- Too vague and unclear

Firewall provides a whole variety of tools necessary for safe and successful exchange of information between hosts in networks

Firewall - history

Note: The term firewall originally referred to a wall intended to confine a fire or potential fire within a building

Firewall - history

Note: The term firewall originally referred to a wall intended to confine a fire or potential fire within a building

The need for firewall emerged in the late 1980s

- Morris Worm in 1988

Firewall – first generation

- First firewall version in 1988 – packet filter

Firewall – first generation

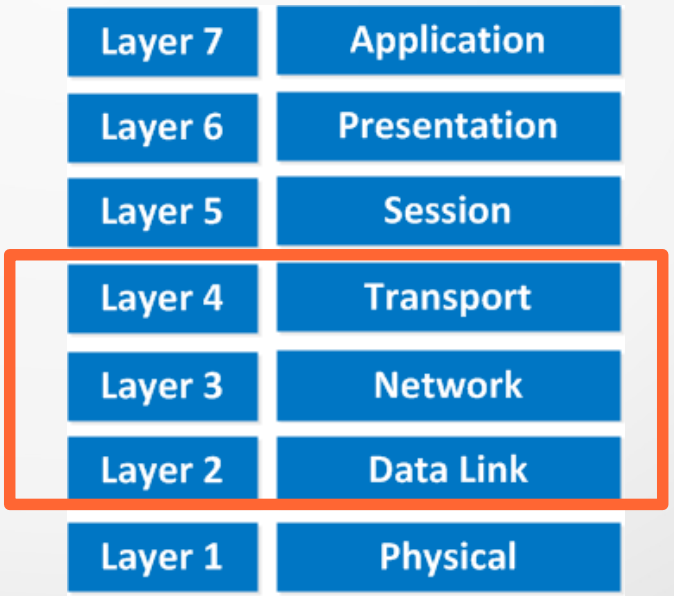
- First firewall version in 1988 – packet filter
- Filters packets based on the contents of packet headers (IP addresses, MAC addresses, protocols (TCP, UDP, ICMP, etc.) and TCP/UDP ports)

Firewall – first generation

- First firewall version in 1988 – packet filter
- Filters packets based on the contents of packet headers (IP addresses, MAC addresses, protocols (TCP, UDP, ICMP, etc.) and TCP/UDP ports)
- Ebtables, arptables

Firewall – first generation

- First firewall version in 1988 – packet filter
- Filters packets based on the contents of packet headers (IP addresses, MAC addresses, protocols (TCP, UDP, ICMP, etc.) and TCP/UDP ports)
- Ebtables, arptables



Firewall – second generation

- Stateful packet filters

Firewall – second generation

- Stateful packet filters
- Packets can be filtered based on their connection state

Firewall – second generation

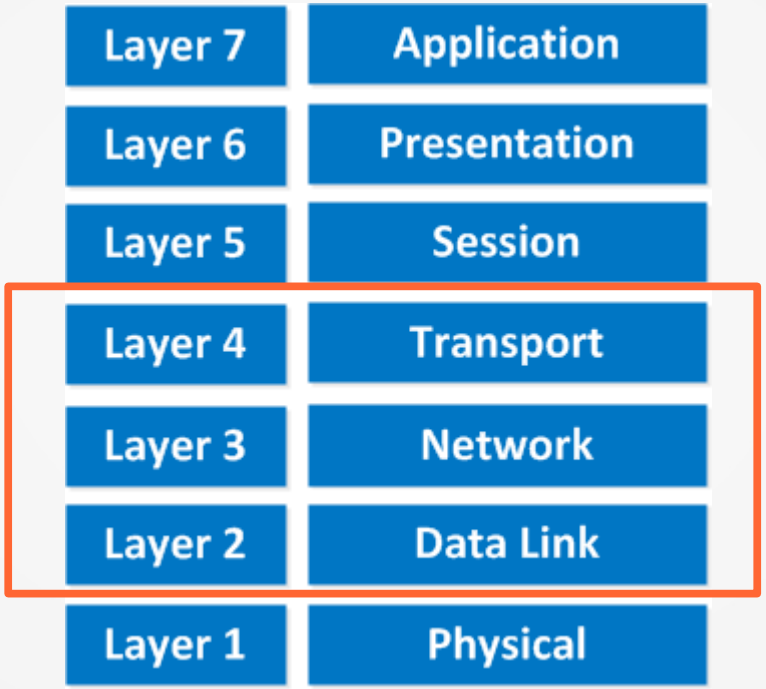
- Stateful packet filters
- Packets can be filtered based on their connection state
- iptables

Firewall – third generation

- Application layer firewall

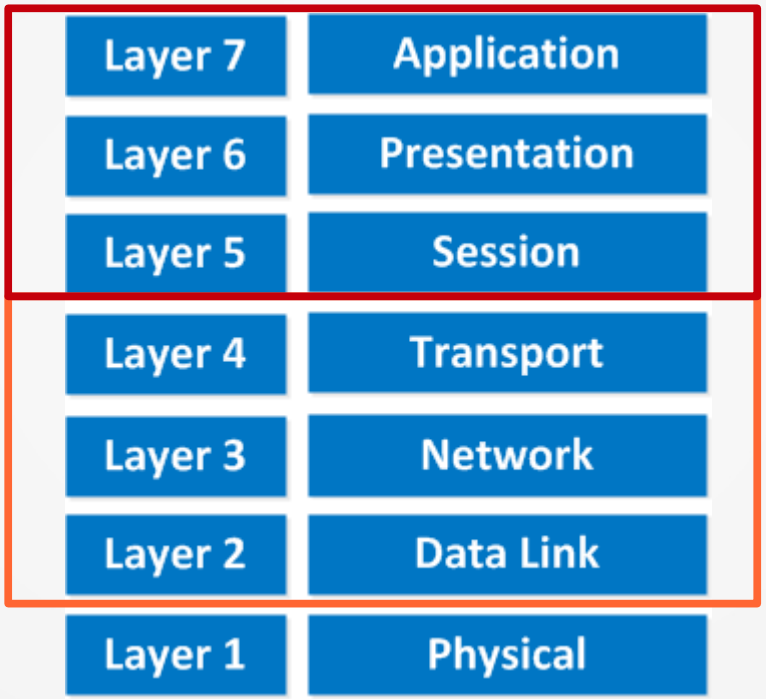
Firewall – third generation

- Application layer firewall



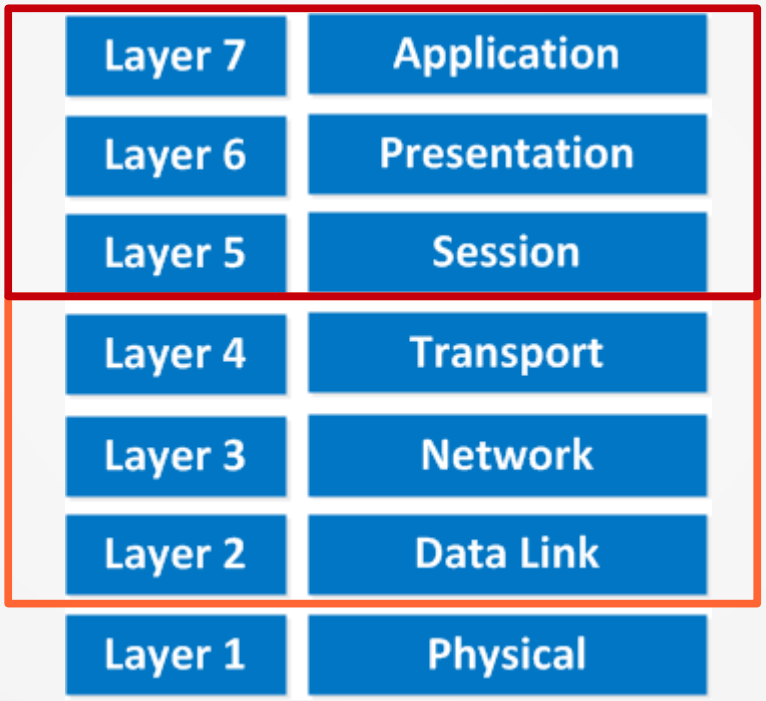
Firewall – third generation

- Application layer firewall



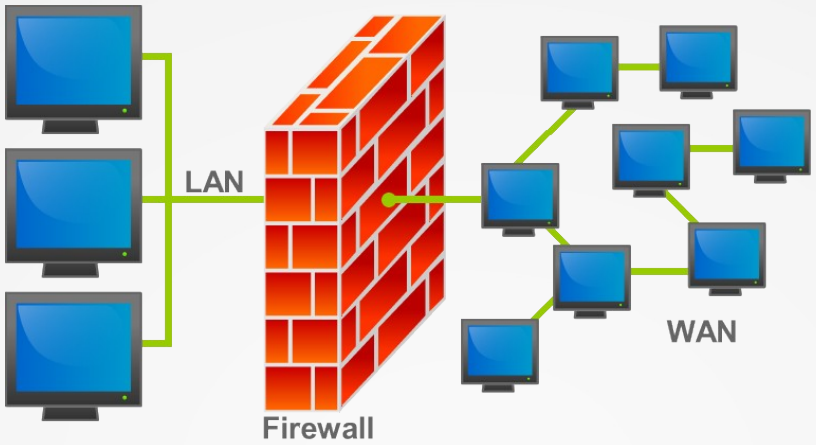
Firewall – third generation

- Application layer firewall



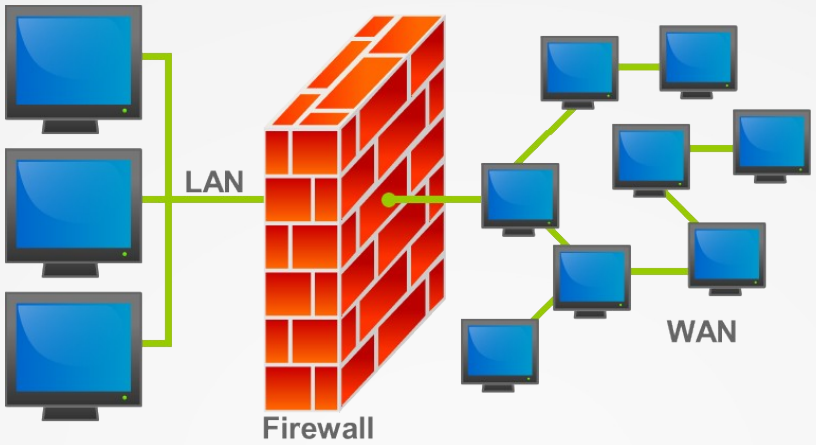
- Able to control network traffic regarding a specific application or sevice
  - AppArmor, Kerio Control

Firewall - concepts



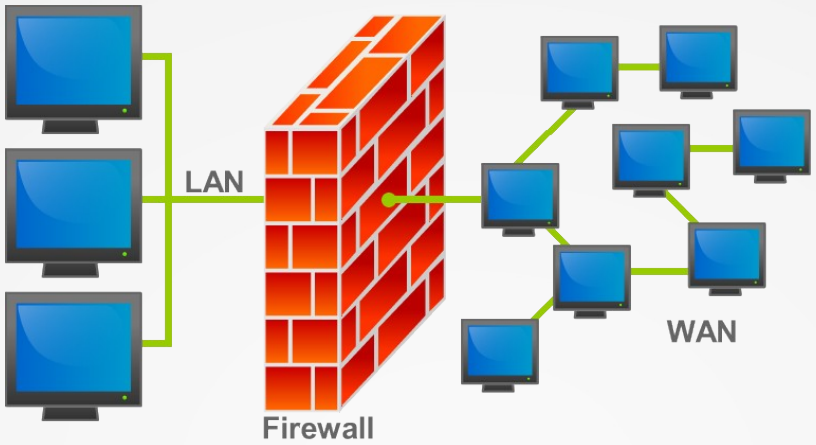
- Firewall consists of 3 main concepts:

Firewall - concepts



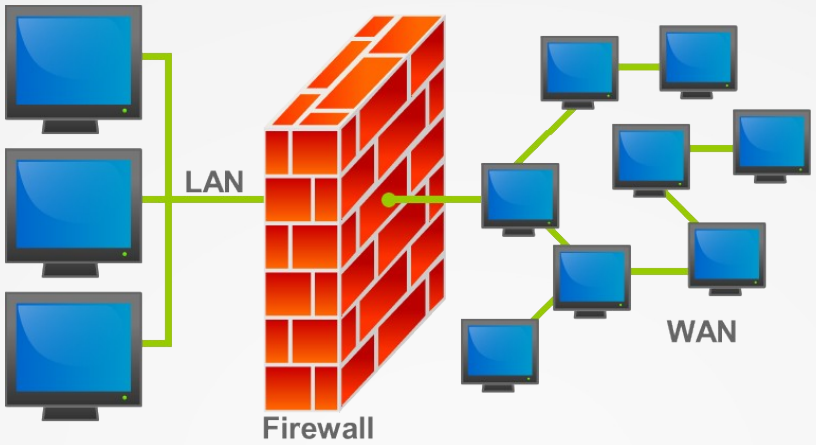
- Firewall consists of 3 main concepts:
  1. Filtering of frames and ARP messages (in OSI layer 2) and packets (in OSI layer 3)

Firewall - concepts



- Firewall consists of 3 main concepts:
  1. Filtering of frames and ARP messages (in OSI layer 2) and packets (in OSI layer 3)
  2. Modification of headers (IP addresses, MAC addresses, ports)

Firewall - concepts



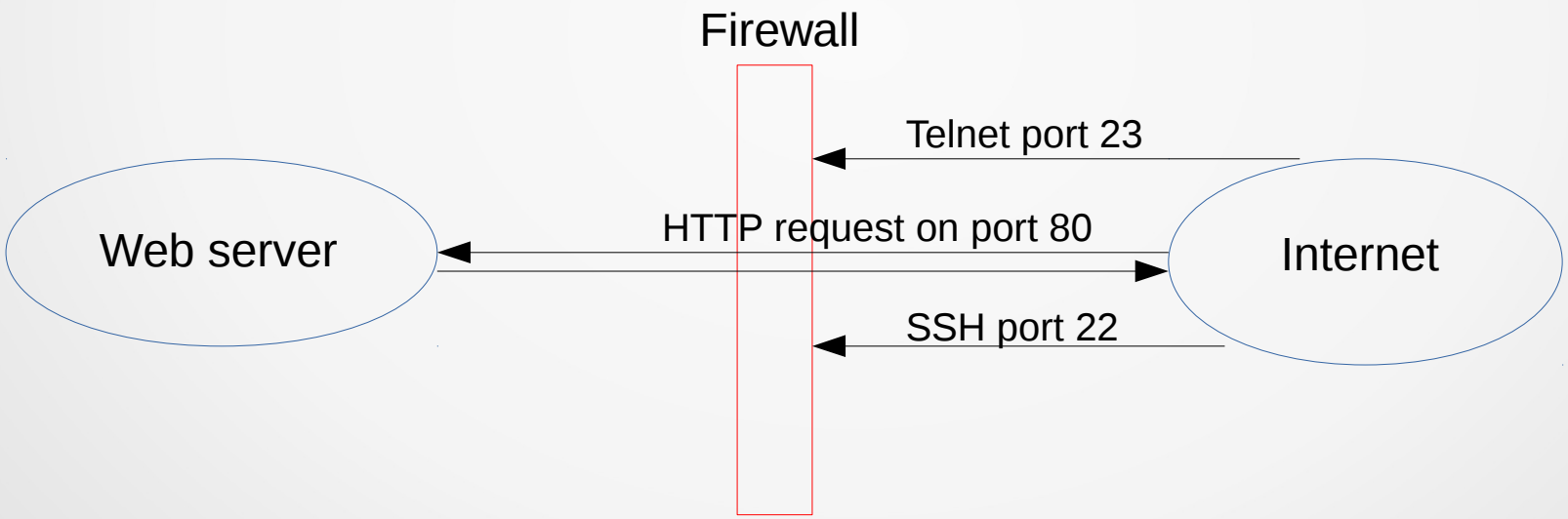
- Firewall consists of 3 main concepts:
  1. Filtering of frames and ARP messages (in OSI layer 2) and packets (in OSI layer 3)
  2. Modification of headers (IP addresses, MAC addresses, ports)
  3. Modification of payload

Firewall – filtering

- The most common use of firewall

Firewall – filtering

- The most common use of firewall
- Example usage: filtering packets based on their destination ports



Firewall - filtering

- Filtering can be done based on:

Firewall - filtering

- Filtering can be done based on:
  - MAC address (OSI layer 2)

Firewall - filtering

- Filtering can be done based on:
  - MAC address (OSI layer 2)
  - IP address (OSI layer 3)

Firewall - filtering

- Filtering can be done based on:
  - MAC address (OSI layer 2)
  - IP address (OSI layer 3)
  - Ports (OSI layer 4)

Firewall - filtering

- Filtering can be done based on:
  - MAC address (OSI layer 2)
  - IP address (OSI layer 3)
  - Ports (OSI layer 4)
  - Protocol (OSI layers 2-4)

Firewall - filtering

- Filtering can be done based on:
  - MAC address (OSI layer 2)
  - IP address (OSI layer 3)
  - Ports (OSI layer 4)
  - Protocol (OSI layers 2-4)
  - Interface

Firewall - filtering

- Filtering can be done based on:
  - MAC address (OSI layer 2)
  - IP address (OSI layer 3)
  - Ports (OSI layer 4)
  - Protocol (OSI layers 2-4)
  - Interface
  - Fragmentation (OSI layer 3)

Firewall - filtering

- Filtering can be done based on:
  - MAC address (OSI layer 2)
  - IP address (OSI layer 3)
  - Ports (OSI layer 4)
  - Protocol (OSI layers 2-4)
  - Interface
  - Fragmentation (OSI layer 3)
  - Payload

Firewall - filtering

- Packet filter can do 3 things with an incoming packet:

Firewall - filtering

- Packet filter can do 3 things with an incoming packet:
  1) Accept it

Firewall - filtering

- Packet filter can do 3 things with an incoming packet:
  1) Accept it
  2) Reject it and notify the sender about rejection

Firewall - filtering

- Packet filter can do 3 things with an incoming packet:
  1) Accept it
  2) Reject it and notify the sender about rejection
  3) Reject it without any notification

Firewall - filtering

- Packet filter can do 3 things with an incoming packet:
  1) Accept it
  2) Reject it and notify the sender about rejection
  3) Reject it without any notification
- Packet filter contains a table with specified set of rules

Firewall - filtering

- Packet filter can do 3 things with an incoming packet:
  1) Accept it
  2) Reject it and notify the sender about rejection
  3) Reject it without any notification
- Packet filter contains a table with specified set of rules
- If incoming packet or frame matches the first rule then the action specified in this rule is performed

Firewall - filtering

- Packet filter can do 3 things with an incoming packet:
  1) Accept it
  2) Reject it and notify the sender about rejection
  3) Reject it without any notification
- Packet filter contains a table with specified set of rules
- If incoming packet or frame matches the first rule then the action specified in this rule is performed
- If it doesn't match, then the next rule is checked (if it exists)

Firewall - filtering

- Packet filter can do 3 things with an incoming packet:
  1) Accept it
  2) Reject it and notify the sender about rejection
  3) Reject it without any notification
- Packet filter contains a table with specified set of rules
- If incoming packet or frame matches the first rule then the action specified in this rule is performed
- If it doesn't match, then the next rule is checked (if it exists)
- If packet or frame doesn't match any of the rules, then is it rejected without any notification

Firewall – header modification

- Firewall can modify information located in headers of packets and frames

Firewall – header modification

- Firewall can modify information located in headers of packets and frames
- Used in IP Masquerading

  IP Masquerading is a process of modifying packet's source IP address

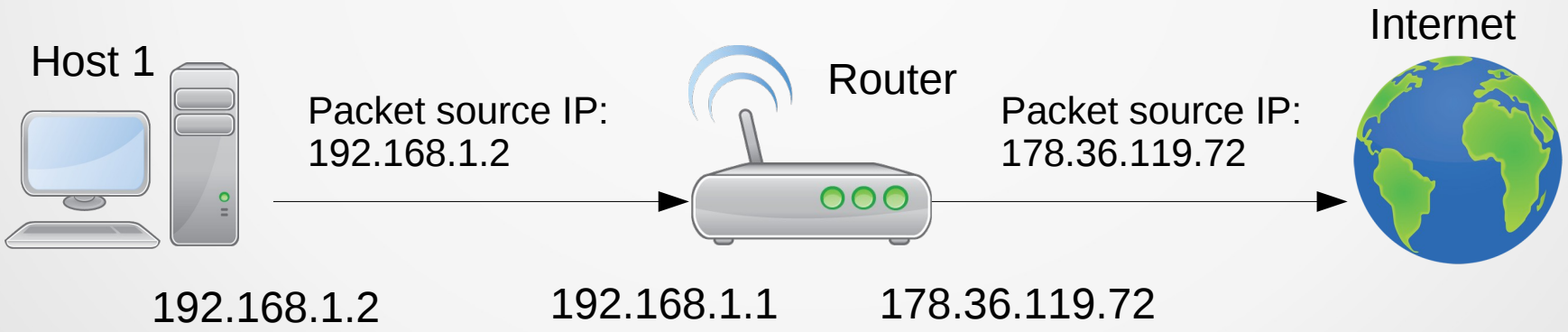
  Used in private networks, where IP Masquerading allows hosts from inside the network that don't have a public IP address to send and receive packets from the Internet.

Firewall – header modification

- Firewall can modify information located in headers of packets and frames
- Used in IP Masquerading

  IP Masquerading is a process of modifying packet's source IP address

  Used in private networks, where IP Masquerading allows hosts from inside the network that don't have a public IP address to send and receive packets from the Internet.

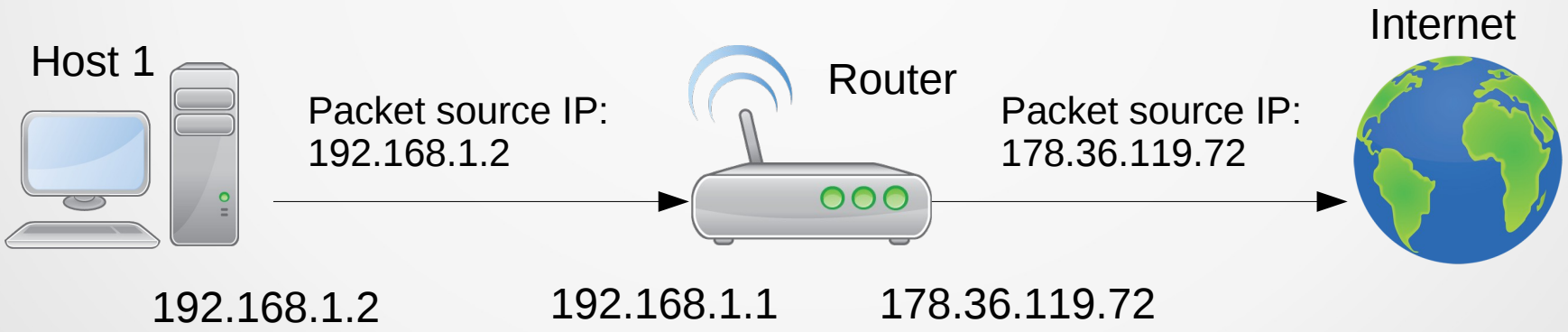


Firewall – header modification

- Firewall can modify information located in headers of packets and frames
- Used in IP Masquerading

  IP Masquerading is a process of modifying packet's source IP address

  Used in private networks, where IP Masquerading allows hosts from inside the network that don't have a public IP address to send and receive packets from the Internet.



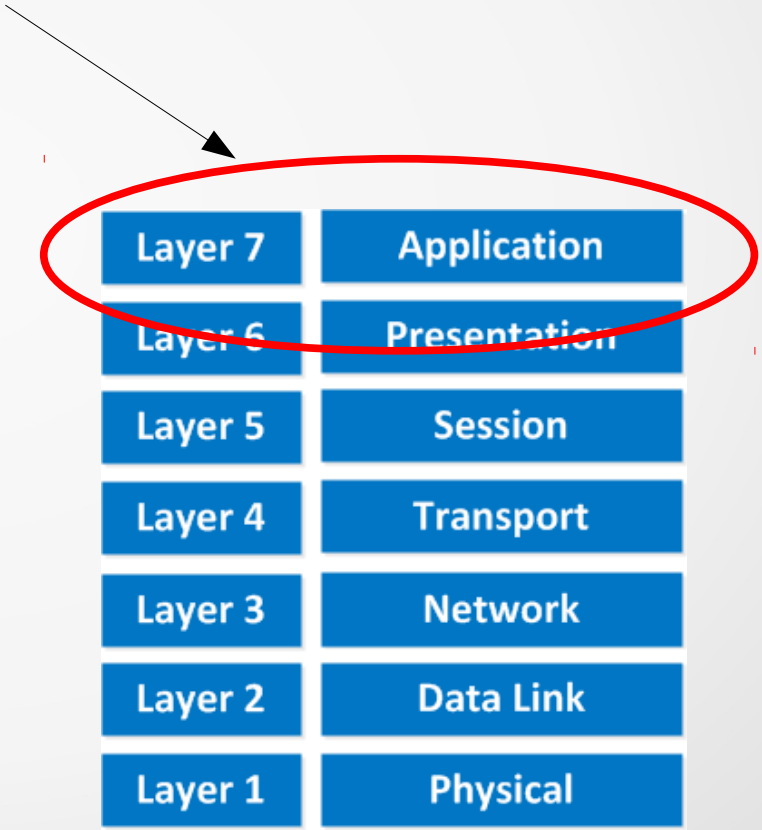
IP Masquerading was performed inside 'Router', where source IP of a packet sent from Host 1 was changed.

Firewall – payload modification

- Frame payload modification takes place mostly in the top layer of OSI model

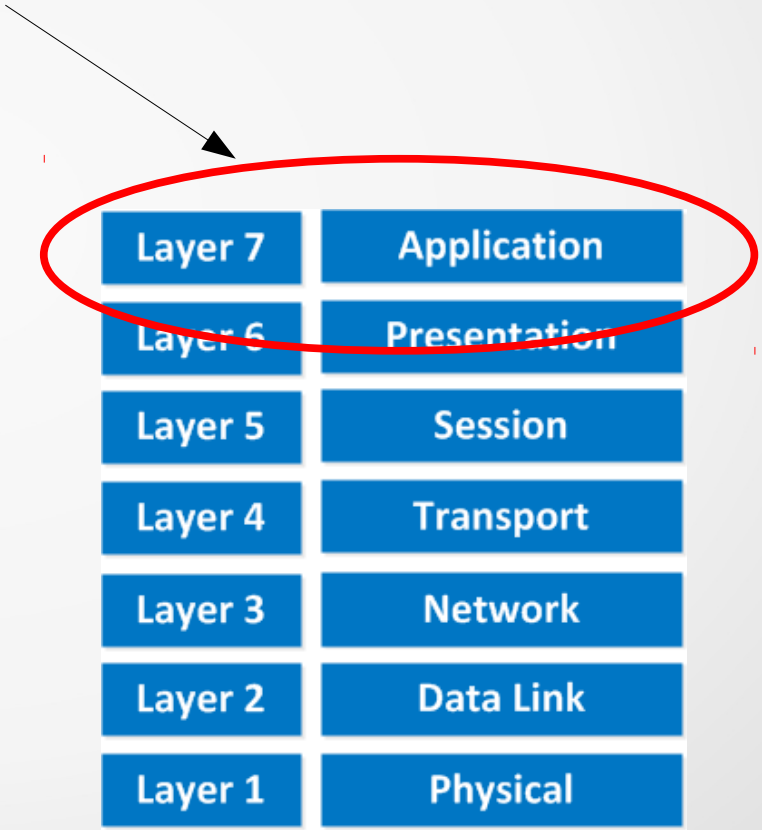
Firewall – payload modification

- Frame payload modification takes place mostly in the top layer of OSI model

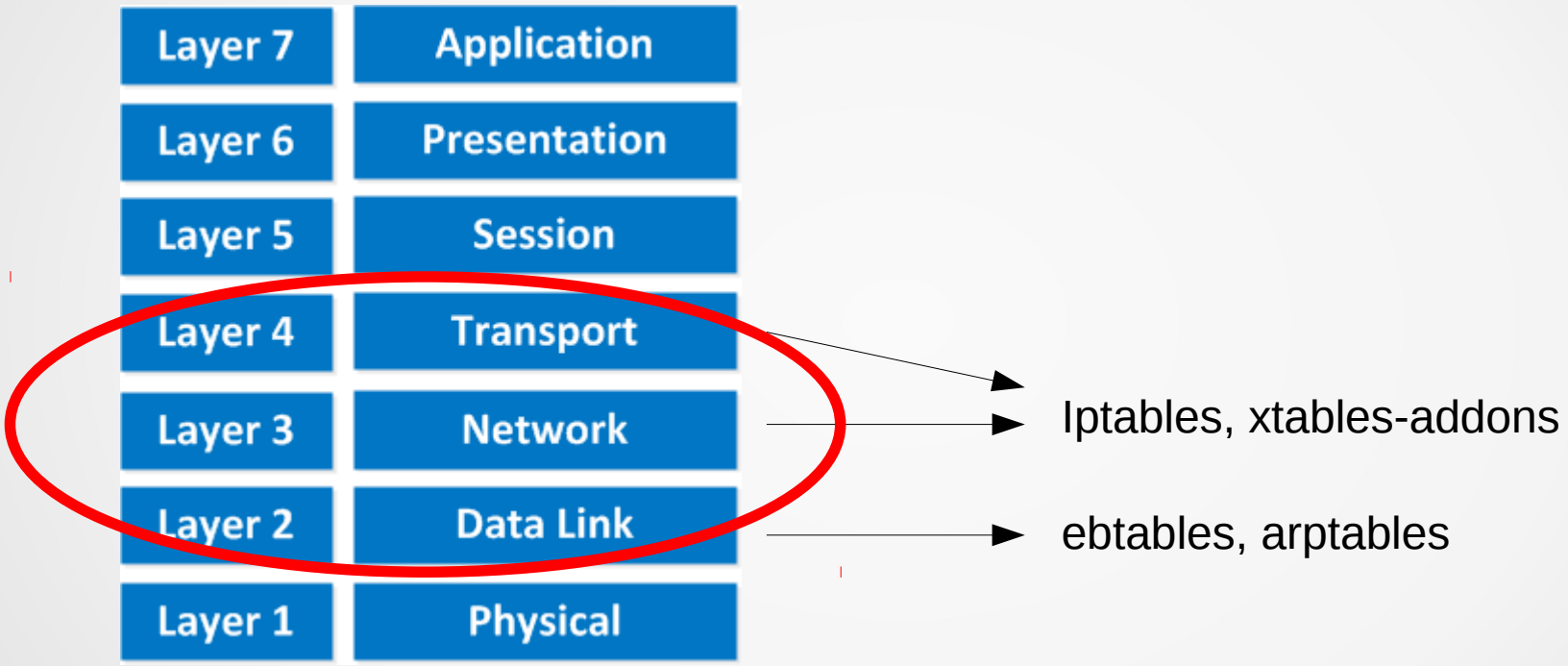


Firewall – payload modification

- Frame payload modification takes place mostly in the top layer of OSI model
- NFQUEUE



Firewall – OSI model



Firewall - ebtables

- The ebtables utility enables basic Ethernet frame filtering on a Linux bridge, logging, MAC NAT and brouting

Firewall - ebtables

- The ebtables utility enables basic Ethernet frame filtering on a Linux bridge, logging, MAC NAT and brouting

- MAC NAT - ability to change the MAC Ethernet source and destination address

Firewall - ebtables

- The ebtables utility enables basic Ethernet frame filtering on a Linux bridge, logging, MAC NAT and brouting

- MAC NAT - ability to change the MAC Ethernet source and destination address
- Brouting – bridge can act as bridge and router. Using firewall rules we can decide if frame should be routed to higher OSI layers or should it be bridged (OSI layer 3)

Firewall – ebtables – tables

Ebtables uses rules to decide what action to perform with frames
Ebtables divides rules into 3 tables:

1. filter
   In this table we place rules which filter frames
2. nat
   In this table we place rules which should modify frame headers
3. broute
   In this table we place rules which should route frames

Rules inside these tables are separated into chains

Firewall – ebtables – tables

Ebtables uses rules to decide what action to perform with frames

Ebtables divides rules into 3 tables:

1. filter

   In this table we place rules which filter frames

   Frames are directed to chain:

   - INPUT - if the destination MAC address of the frame is on the bridge itself
   - FORWARD - for frames being forwared by the bridge
   - OUTPUT - if the frames are generated locally or for brouted frames

2. nat

   In this table we place rules which should modify frame headers

   - PREROUTING – frames are modified as soon as they came in
   - OUTPUT - for modifying locally generated or (b)routed frames before they are bridged
   - POSTROUTING - for altering frames as they are about to go out

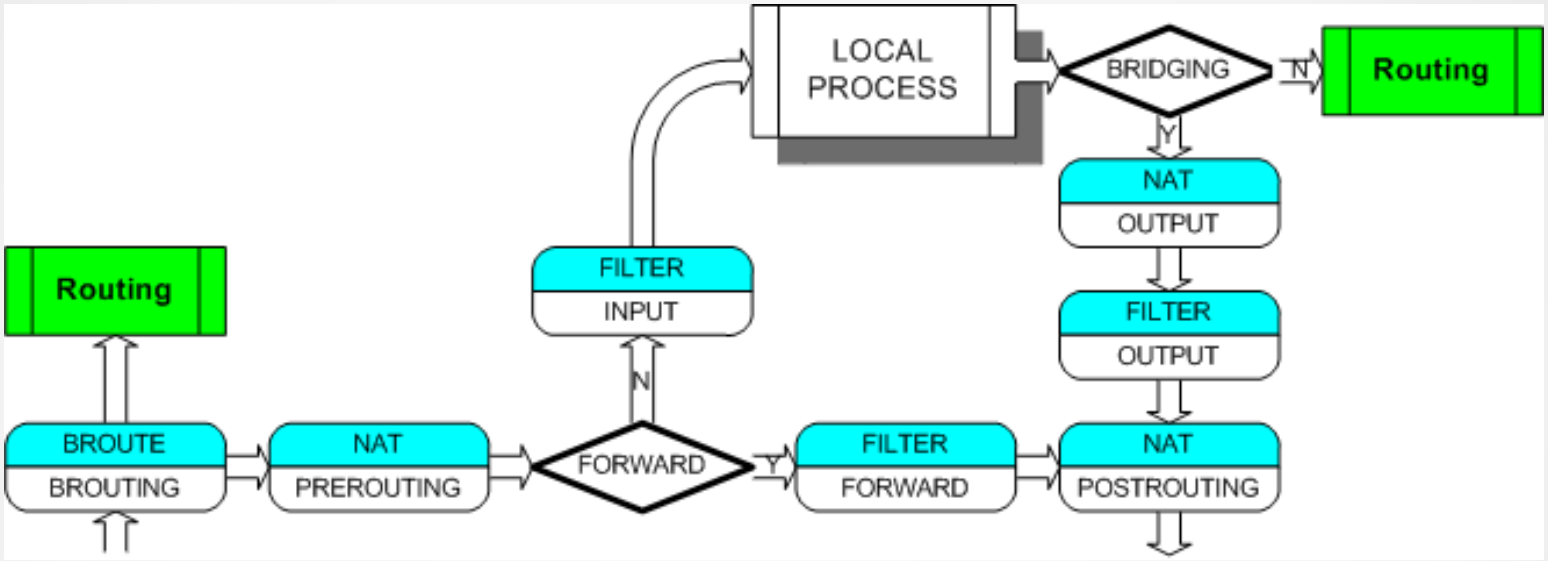
3. broute

   In this table we place rules which should route frames

   - BROUTING

Users can also define their own chains

Firewall – ebtables – tables



Firewall – ebtables - bridge

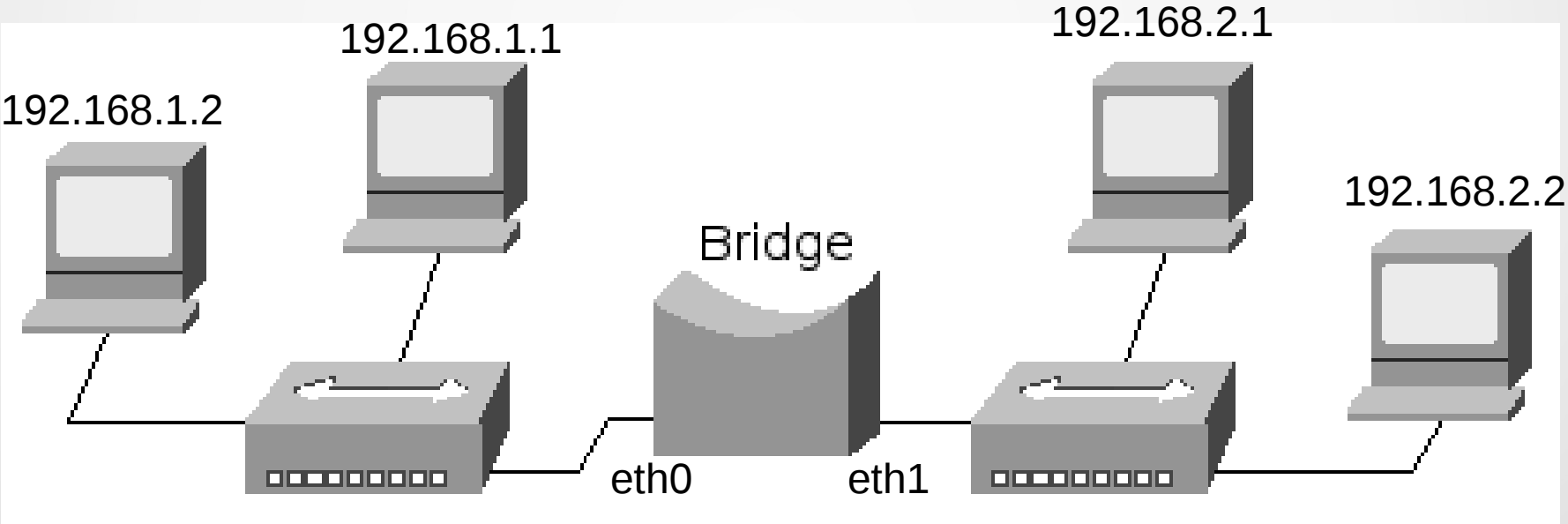
- A network bridge connects multiple network segments

Firewall – ebtables - bridge

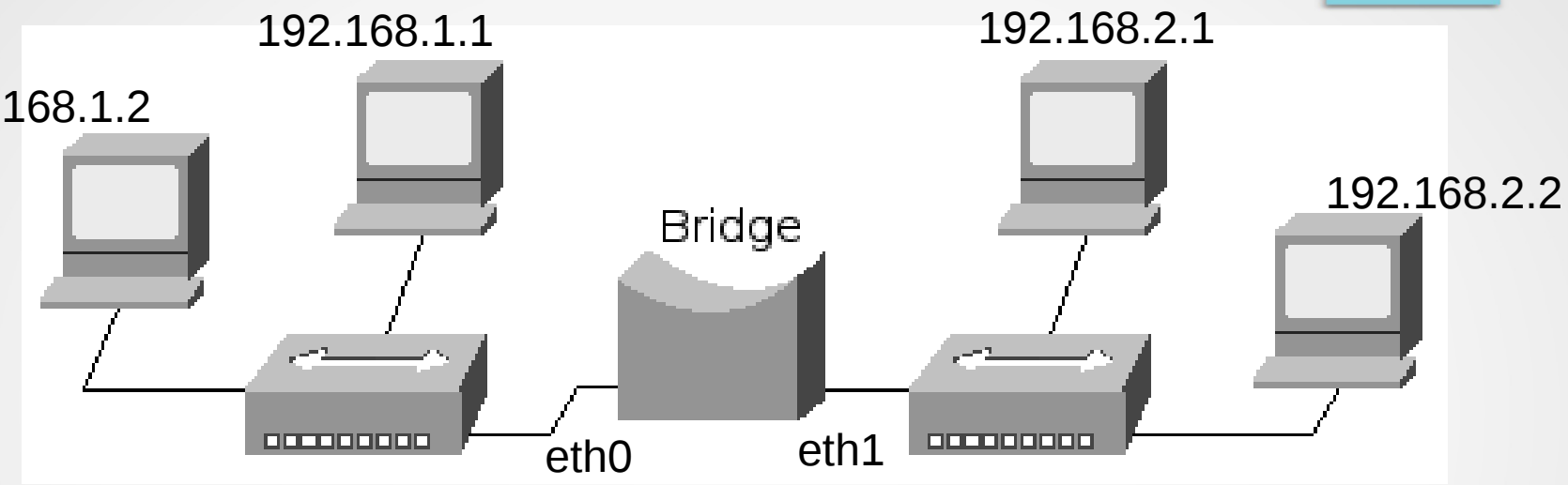
- A network bridge connects multiple network segments
- To create a bridge on a computer we will use *brctl* located in *bridge-utils* package

Firewall – ebtables - bridge

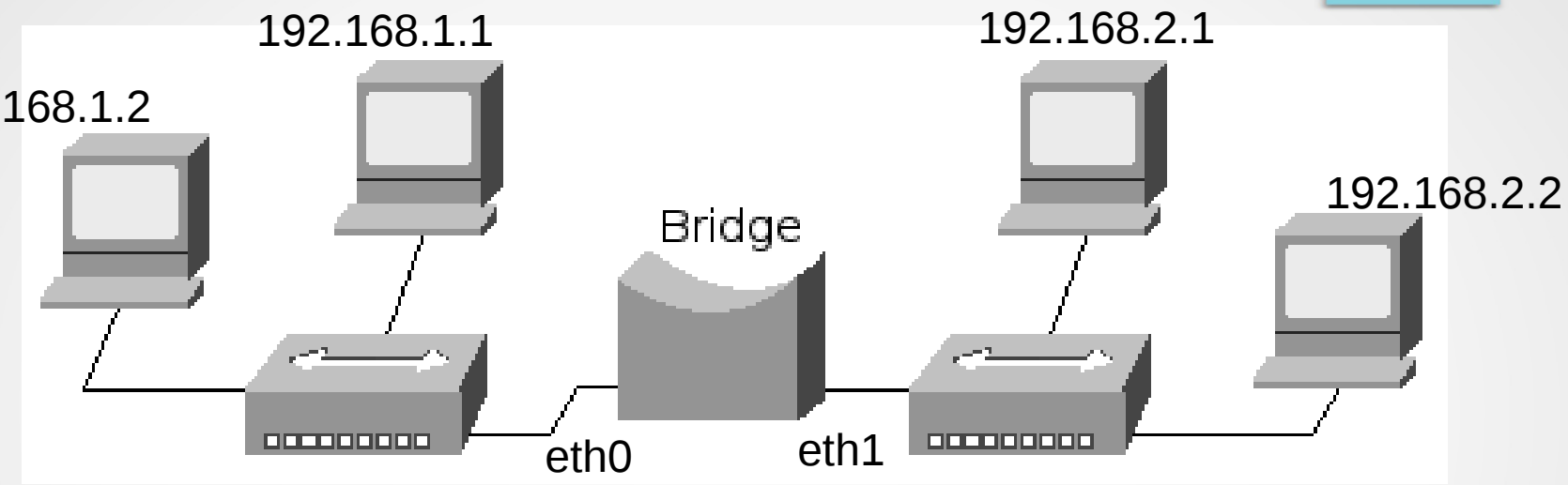
- A network bridge connects multiple network segments
- To create a bridge on a computer we will use *brctl* located in *bridge-utils* package



Firewall – ebtables - bridge

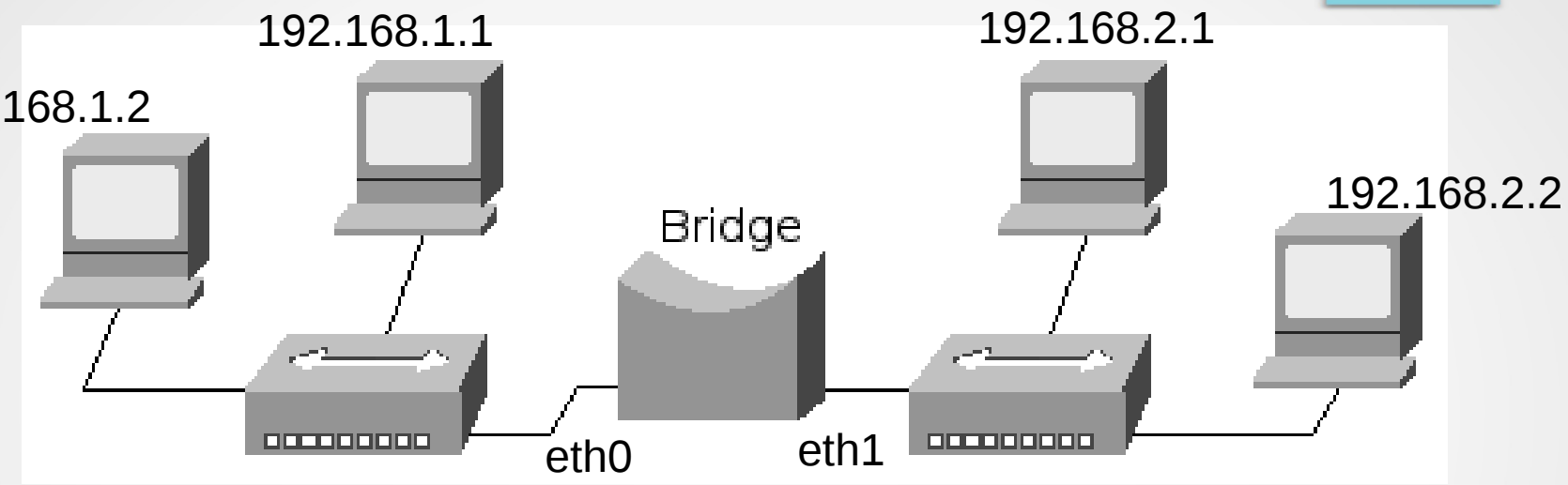


Firewall – ebtables - bridge




```
root@Bridge:# ifconfig eth0 0.0.0.0
root@Bridge:# ifconfig eth1 0.0.0.0
root@Bridge:# brctl addbr br0
root@Bridge:# brctl addif br0 eth0
root@Bridge:# brctl addif br0 eth1
root@Bridge:# ifconfig br0 up
```

Firewall – ebtables - bridge




```
root@Bridge:# ifconfig eth0 0.0.0.0
root@Bridge:# ifconfig eth1 0.0.0.0
root@Bridge:# brctl addbr br0
root@Bridge:# brctl addif br0 eth0
root@Bridge:# brctl addif br0 eth1
root@Bridge:# ifconfig br0 up
```

```
root@Bridge:# brctl showmacs br0
port no mac addr                is local?       ageing timer
  1     00:10:4b:b6:c6:e4       no              119.25
  1     00:a0:24:d0:4c:d6       yes               0.00
  1     00:a0:24:f0:22:71       no                5.81
  4     08:00:09:fb:39:a1       no               27.24
  4     08:00:09:fc:92:2c       no               53.13
  4     08:00:09:fc:d2:11       yes               0.00
```

Firewall – ebtables – installation & rules

Installation:

```
root@Bridge:# apt-get install ebtables
```

Firewall – ebtables – installation & rules

Installation:

```
root@Bridge:# apt-get install ebtables
```

- Ebtables holds a table of rules, which are used to filter frames

Firewall – ebtables – installation & rules

Installation:

root@Bridge:# apt-get install ebtables

- Ebtables holds a table of rules, which are used to filter frames

To list these rules we can use a command:

root@Bridge:# ebtables -L

Firewall – ebtables – installation & rules

Installation:

root@Bridge:# apt-get install ebtables

- Ebtables holds a table of rules, which are used to filter frames

To list these rules we can use a command:

root@Bridge:# ebtables -L

```
Bridge table: filter

Bridge chain: INPUT, entries: 0, policy: ACCEPT

Bridge chain: FORWARD, entries: 0, policy: ACCEPT

Bridge chain: OUTPUT, entries: 0, policy: ACCEPT
```

Firewall – ebtables – commands

Append the rule to the end of the chain: iptables -A *rule_specification* -j *target*

root@Harvey:# iptables -A INPUT --dport 80 -j DROP

List rules in chain: iptables -L [chain]

root@Harvey:# iptables -L

Flush (delete all) rules in chain: iptables -F *chain*

root@Harvey:# iptables -F OUTPUT

To delete a rule: ebtables -D *chain start_nr*[:*end_nr*]

root@Bridge:# ebtables -D FORWARD 1

To create a user-defined chain: ebtables -N *chain_name [-P policy]*

root@Bridge*:*# ebtables -D new_chain -P ACCEPT

To specify the destination MAC address: ebtables -A *chain* -s *mac_address* -j *TARGET*

root@Bridge:# ebtables -A FORWARD -s *b8:88:e3:79:1a:b1* -j DROP

To specify the source MAC address: ebtables -A *chain* -s *mac_address* -j *TARGET*

root@Bridge:# ebtables -A FORWARD -s *b8:88:e3:79:1a:b1* -j DROP

Firewall – ebtables – rule specifications (matches)

Rule specifications:

-p *protocol* - protocol responsible for creating the frame

-i *interface* - the interface the frame was originally received from

-o *interface* - the interface by which the frame is going to be sent

-s *source* - source MAC address

-d *destination* - destination MAC address

Firewall – ebtables – targets

TARGET specifies what happens to a frame after it matched a rule.

Firewall – ebtables – targets

TARGET specifies what happens to a frame after it matched a rule.

In Ebtables we can choose from 5 targets:
1) ACCEPT – let the frame through
2) DROP – the frame will be dropped
3) CONTINUE – next rule will be checked
4) RETURN – go back to the previous chain and check the next rule
5) Jump to user defined chain

Firewall – ebtables – basic configuration

ebtables -P FORWARD DROP
ebtables -A FORWARD -p IPv4 -j ACCEPT
ebtables -A FORWARD -p ARP -j ACCEPT
ebtables -P INPUT DROP
ebtables -A INPUT -p IPv4 -j ACCEPT
ebtables -A INPUT -p ARP -j ACCEPT
ebtables -P OUTPUT DROP
ebtables -A OUTPUT -p IPv4 -j ACCEPT
ebtables -A OUTPUT -p ARP -j ACCEPT

This is a basic filter configuration which will only let frames made by the protocols IP version 4 and ARP through

Firewall – ebtables – MAC NAT

- Ebtables allows for changing MAC addresses in frame headers

Firewall – ebtables – MAC NAT

- Ebtables allows for changing MAC addresses in frame headers

root@Bridge:# ebtables -t nat -A PREROUTING -d 00:11:22:33:44:55 -i eth0 -j dnat --to-destination 54:44:33:22:11:00

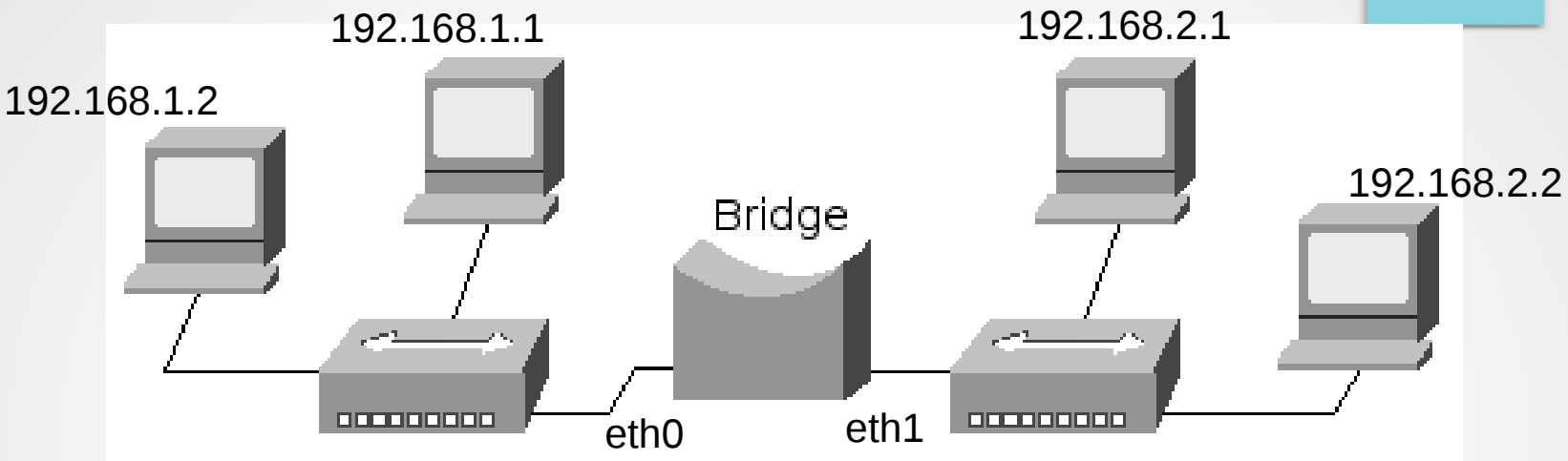
Firewall – ebtables – MAC NAT

- Ebtables allows for changing MAC addresses in frame headers

root@Bridge:# ebtables -t nat -A PREROUTING -d 00:11:22:33:44:55 -i eth0 -j dnat --to-destination 54:44:33:22:11:00

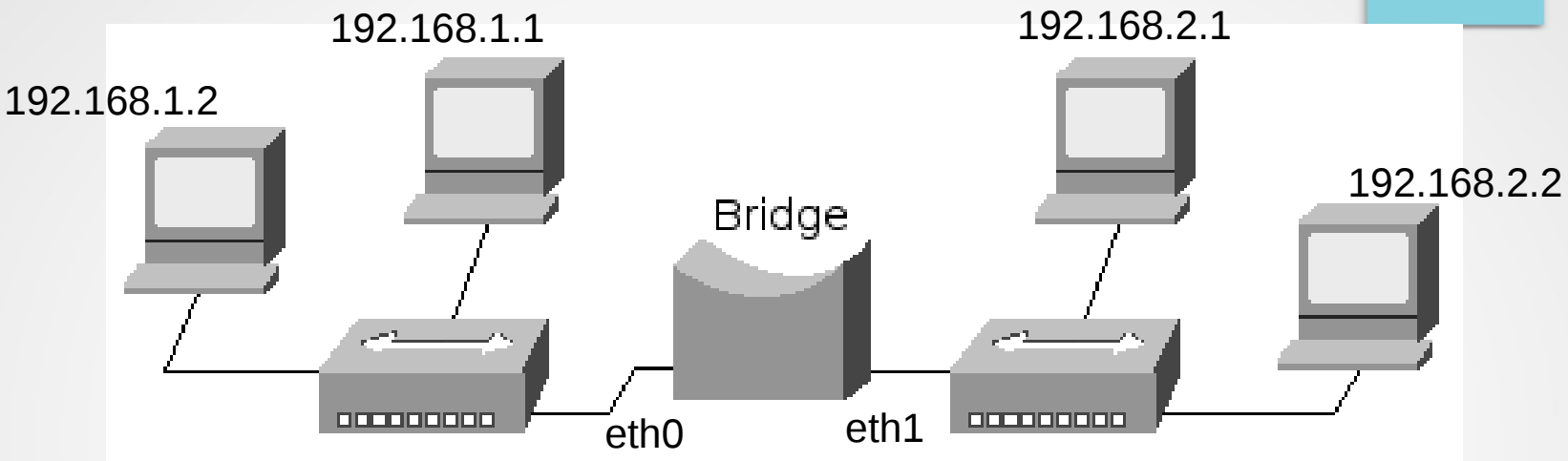
This will make all frames destined to 00:11:22:33:44:55 that arrived on interface eth0 be transferred to 54:44:33:22:11:00 instead

Firewall – ebtables - example



- Example: we want to block ARP messages going through the bridge

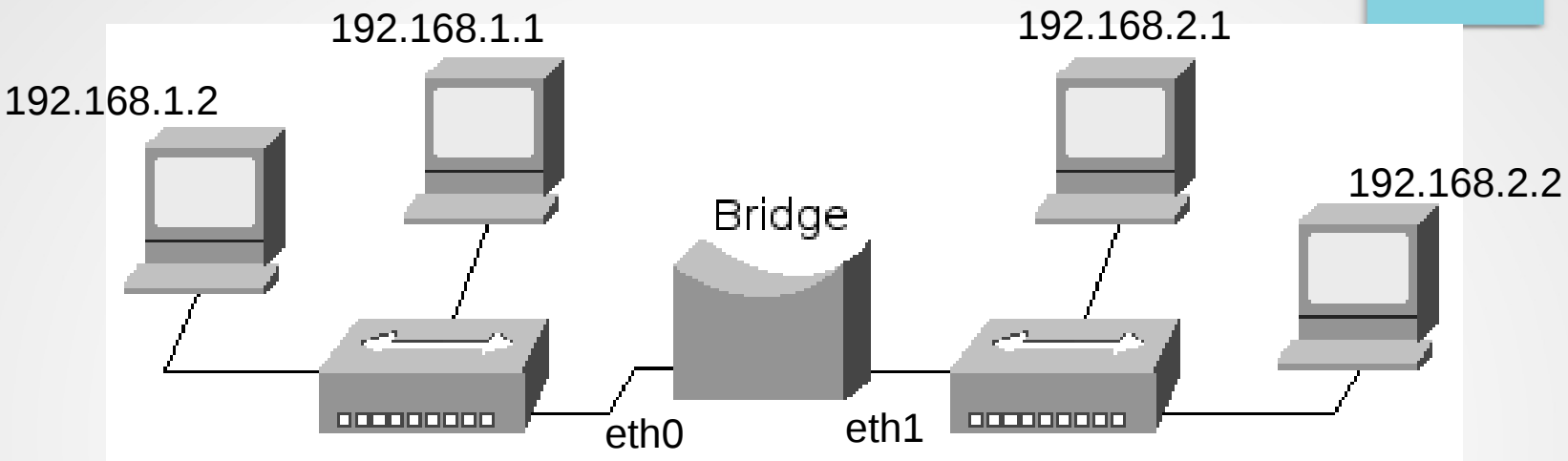
Firewall – ebtables - example



- Example: we want to block ARP messages going through the bridge

```
root@192.168.1.2:# arping 192.168.2.1
ARPING 192.168.2.1 from 192.168.1.2 eth0
Unicast reply from 192.168.2.1 [B8:54:E5:34:1A:E4]  0.950ms
Unicast reply from 192.168.2.1 [B8:54:E5:34:1A:E4]  1.055ms
```

Firewall – ebtables - example

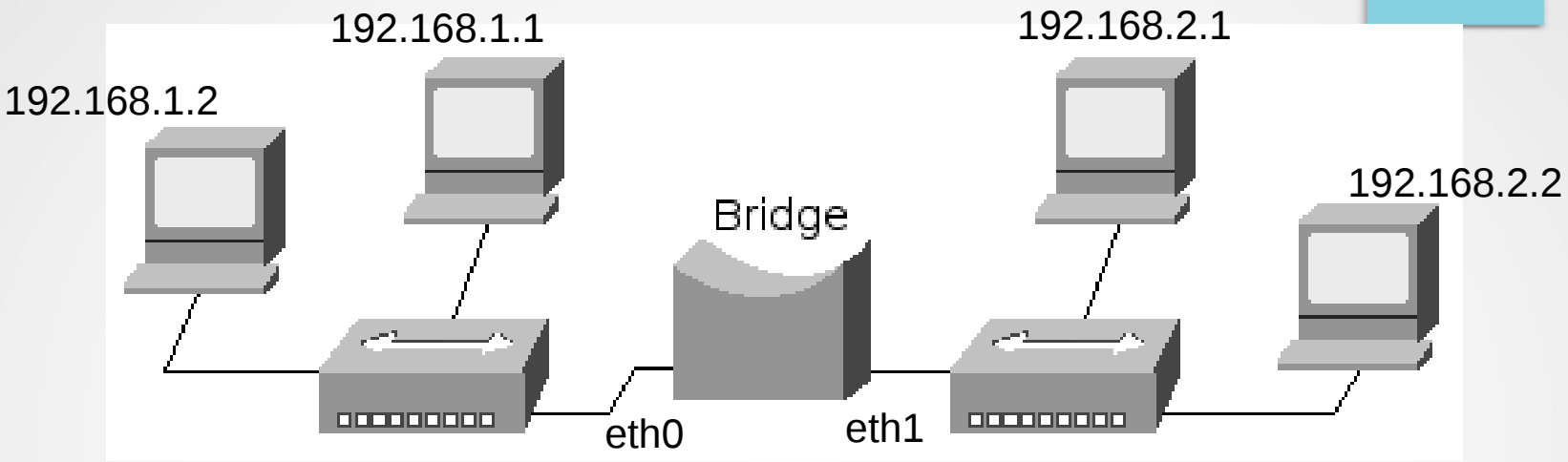


- Example: we want to block ARP messages going through the bridge

root@192.168.1.2:# arping 192.168.2.1
ARPING 192.168.2.1 from 192.168.1.2 eth0
Unicast reply from 192.168.2.1 [B8:54:E5:34:1A:E4] 0.950ms
Unicast reply from 192.168.2.1 [B8:54:E5:34:1A:E4] 1.055ms

root@Bridge:# ebtables -t filter -A FORWARD -p ARP -j DROP

Firewall – ebtables - example



- Example: we want to block ARP messages going through the bridge

```
root@192.168.1.2:# arping 192.168.2.1
ARPING 192.168.2.1 from 192.168.1.2 eth0
Unicast reply from 192.168.2.1 [B8:54:E5:34:1A:E4]  0.950ms
Unicast reply from 192.168.2.1 [B8:54:E5:34:1A:E4]  1.055ms

root@Bridge:# ebtables -t filter -A FORWARD -p ARP -j DROP

root@192.168.1.2:# arping 192.168.2.1 -c 5
ARPING 192.168.2.1 from 192.168.1.2 eth0
Sent 5 probes (5 broadcast(s))
Received 0 response(s)
```

Firewall - arptables

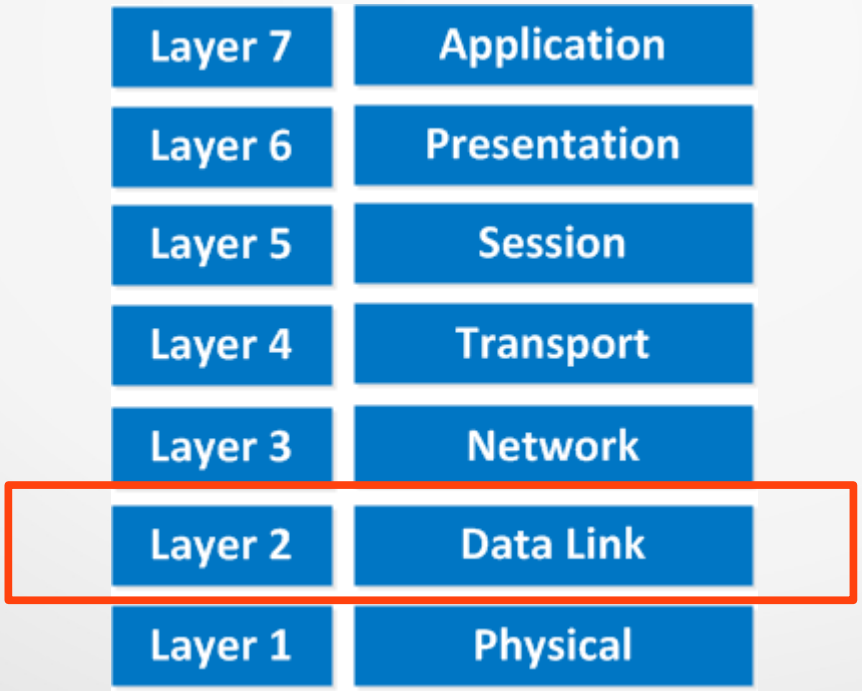
- Configuration very similar to ebtables

Firewall - arptables

- Configuration very similar to ebtables
- Not limited to bridges
- Designed to work specifically with ARP messages

Firewall - arptables

- Configuration very similar to ebtables
- Not limited to bridges
- Designed to work specifically with ARP messages



Firewall – arptables – tables

Installation:

root@Harvey:# apt-get install arptables

Firewall – arptables – tables

Installation:

root@Harvey:# apt-get install arptables

```
root@Harvey:/home/arclite# arptables -L
Chain INPUT (policy ACCEPT)

Chain OUTPUT (policy ACCEPT)

Chain FORWARD (policy ACCEPT)
```

Firewall – arptables – tables

Installation:

root@Harvey:# apt-get install arptables

```
root@Harvey:/home/arclite# arptables -L
Chain INPUT (policy ACCEPT)

Chain OUTPUT (policy ACCEPT)

Chain FORWARD (policy ACCEPT)
```

- One *filter* table
  1) INPUT chain - for frames destined for the host
  2) OUTPUT chain - for locally-generated frames
  3) FORWARD chain - for frames being forwarded by the bridge (this chain is not available in Linux kernel 2.4.X)

Firewall – arptables – rule building & commands

Rule building:

arptables [-t table] command rule-specification [options]

- Option -t can be omitted

To add a new rule to a specific chain:
arptables -A *chain rule_specification*

To delete a rule from chain:
arptables -D *chain rule_specification*
or
arptables -D *chain start_nr[:end_nr]*

To set the policy for the chain: (only for built-in chains)
arptables -P *chain policy* (ACCEPT/DROP/RETURN)

To add a new user-defined chain:
arptables -N *chain_name*

Firewall – arptables – rule building & commands

Rule building:

arptables [-t table] command rule-specification [options]

- Option -t can be omitted

Firewall – arptables – rule building & commands

Rule building:

arptables [-t table] command rule-specification [options]

- Option -t can be omitted

To add a new rule to a specific chain:

arptables -A *chain rule_specification*

Firewall – arptables – rule building & commands

Rule building:

arptables [-t table] command rule-specification [options]

- Option -t can be omitted

To add a new rule to a specific chain:

arptables -A *chain rule_specification*

To delete a rule from chain:

arptables -D *chain rule_specification*

or

arptables -D *chain start_nr[:end_nr]*

Firewall – arptables – rule building & commands

Rule building:

arptables [-t table] command rule-specification [options]

- Option -t can be omitted

To add a new rule to a specific chain:
arptables -A *chain rule_specification*

To delete a rule from chain:
arptables -D *chain rule_specification*
or
arptables -D *chain start_nr[:end_nr]*

To set the policy for the chain: (only for built-in chains)
arptables -P *chain policy* (ACCEPT/DROP/RETURN)

Firewall – arptables – rule building & commands

Rule building:

arptables [-t table] command rule-specification [options]

- Option -t can be omitted

To add a new rule to a specific chain:
arptables -A *chain rule_specification*

To delete a rule from chain:
arptables -D *chain rule_specification*
or
arptables -D *chain start_nr[:end_nr]*

To set the policy for the chain: (only for built-in chains)
arptables -P *chain policy* (ACCEPT/DROP/RETURN)

To add a new user-defined chain:
arptables -N *chain_name*

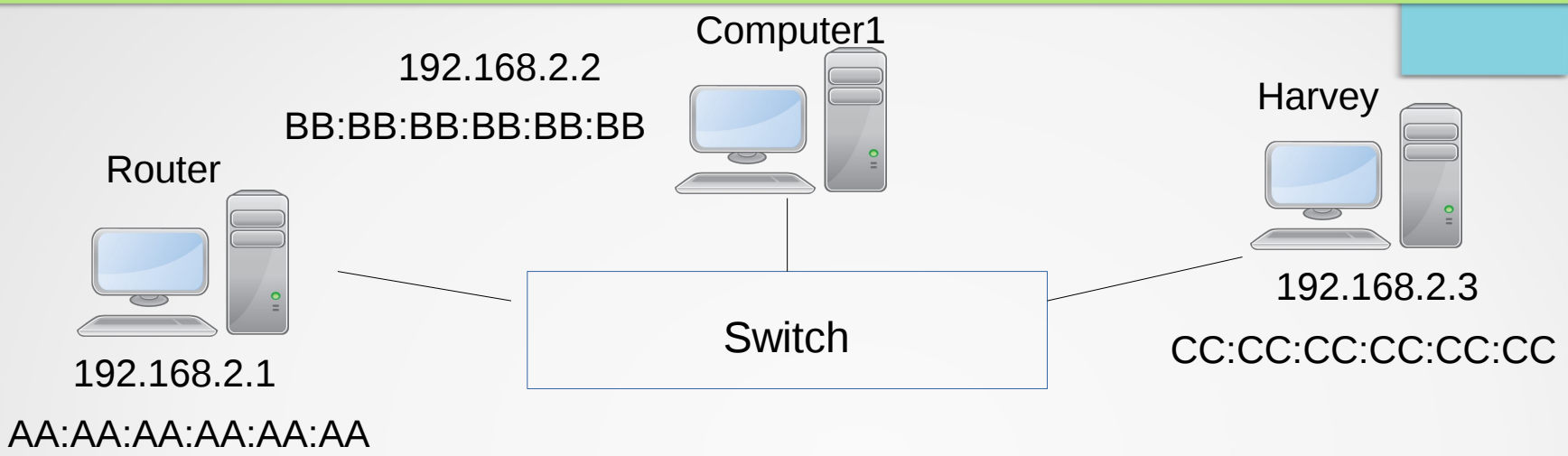
Firewall – arptables – rule specifications

Rule specifications:

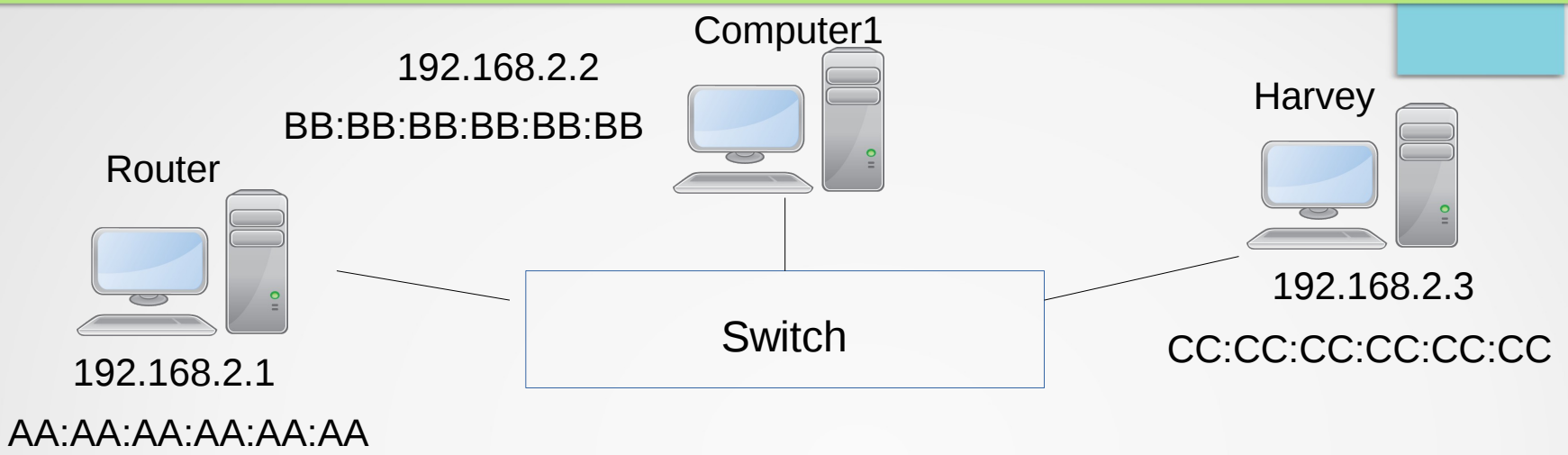
- '-s *address[/mask]*' - the Source IP specification
- '-d *address[/mask]*' - the Destination IP specification
- *'*--source-mac *mac_address'* - the source mac address
- *'*--destination-mac *mac_address'* - the destination mac address
- *'*-i *name'* - the interface via which a frame is received (for the INPUT and FORWARD chains)
- '-o *name'* - the interface via which a frame is going to be sent (for the OUTPUT and FORWARD chains)

'!' option before the specification inverts the test for that specification

Firewall - arptables - example

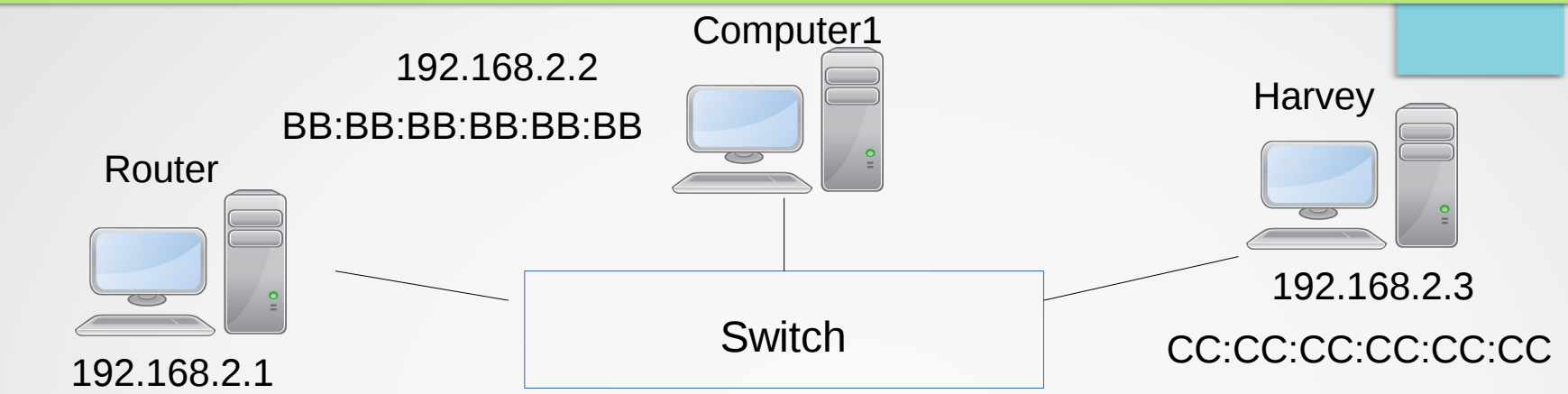


Firewall - arptables - example



- Harvey sends valuable data to Router and wants to protect himself from Computer1 ARP Spoofing

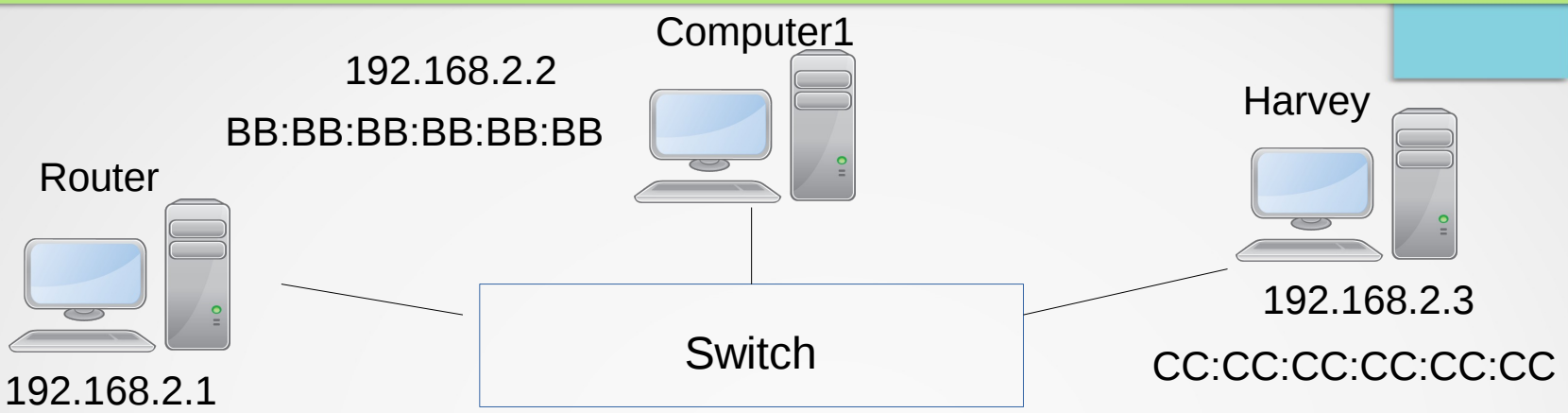
Firewall - arptables - example



- Harvey sends valuable data to Router and wants to protect himself from Computer1 ARP Spoofing

root@Harvey:# arptables -A INPUT -s 192.168.2.1 --source-mac ! AA:AA:AA:AA:AA:AA -j DROP
root@Harvey:# arptables -A OUTPUT -d 192.168.2.1 --destination-mac ! AA:AA:AA:AA:AA:AA -j DROP

Firewall - arptables - example

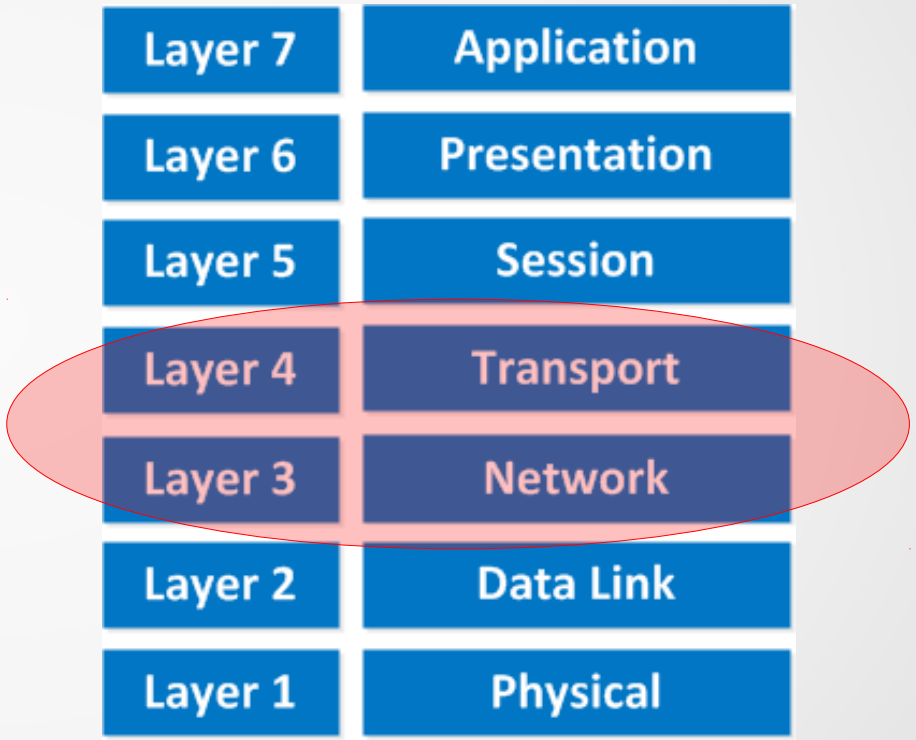


- Harvey sends valuable data to Router and wants to protect himself from Computer1 ARP Spoofing

```
root@Harvey:# arptables -A INPUT -s 192.168.2.1 --source-mac ! AA:AA:AA:AA:AA:AA -j DROP
root@Harvey:# arptables -A OUTPUT -d 192.168.2.1 --destination-mac ! AA:AA:AA:AA:AA:AA -j
DROP
root@Harvey:# arptables -L
Chain INPUT (policy ACCEPT)
-j DROP -s 192.168.2.1 ! --src-mac aa:aa:aa:aa:aa:aa

Chain OUTPUT (policy ACCEPT)
-j DROP -d 192.168.2.1 ! --dst-mac aa:aa:aa:aa:aa:aa

Chain FORWARD (policy ACCEPT)
```

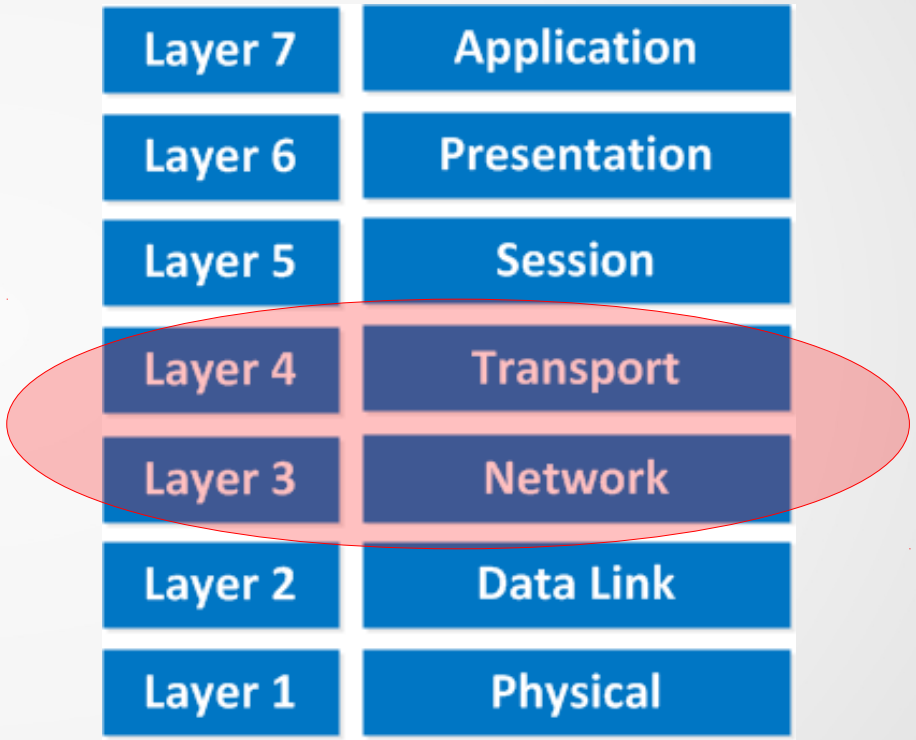
Firewall - iptables

- Iptables is a firewall for OSI layer 3 and 4



Firewall - iptables

- Iptables is a firewall for OSI layer 3 and 4
- successor of ipchains and ipfilter



Firewall - iptables - tables

- filter
  - INPUT
  - FORWARD
  - OUTPUT
- nat
  - PREROUTING
  - OUTPUT
  - POSTROUTING
- mangle
  - PREROUTING
  - INPUT
  - OUTPUT
  - FORWARD
  - POSTROUTING
- raw
  - PREROUTING
  - OUTPUT

Firewall - iptables - tables

- mangle - This table is used for specialized packet modification (e.g. change Type Of Service field in a packet)
- raw - handled before connection tracking takes place

Firewall - iptables - rule specifications (matches)

- Other matches:
  - -m or --match
    - -m conntrack
      Allows filter rules to match based on connection state. Permits the use of the --ctstate option.

    - --ctstate
      Example: iptables -A INPUT -p tcp -m conntrack --ctstate NEW -j ACCEPT

Firewall - iptables - rule specifications (matches)

- Other matches:
  - -m or --match
    - -m conntrack
      Allows filter rules to match based on connection state. Permits the use of the --ctstate option.

    - --ctstate
      Example: iptables -A INPUT -p tcp -m conntrack --ctstate NEW -j ACCEPT

    - -m limit - Require the rule to match only a limited number of times. Allows the use of the --limit option. Useful for limiting logging rules.

      - --limit and --limit-burst
        Example: iptables -A INPUT -p icmp -m limit --limit 1/min --limit-burst 5 -j DROP

Firewall - iptables - features

- stateless packet filtering (IPv4 and IPv6)

Firewall - iptables - features

- stateless packet filtering (IPv4 and IPv6)
- stateful packet filtering (IPv4 and IPv6)
  - iptables can distinguish the connection state of a packet
  - this adds a new packet filtering capability based on the connection state

Firewall - iptables - features

- stateless packet filtering (IPv4 and IPv6)
- stateful packet filtering (IPv4 and IPv6)
  - iptables can distinguish the connection state of a packet
  - this adds a new packet filtering capability based on the connection state
- NAPT - network address and port translation (IPv4 and IPv6)
  - iptables can modify packet headers including IP addresses and ports

Firewall - iptables - connection state

- Packets can be related to tracked connections in four different so called states: NEW, ESTABLISHED, RELATED and INVALID

Firewall - iptables - connection state

- Packets can be related to tracked connections in four different so called states: NEW, ESTABLISHED, RELATED and INVALID
- NEW - tells us that the packet is the first packet that we see from a specific connection

Firewall - iptables - connection state

- Packets can be related to tracked connections in four different so called states: NEW, ESTABLISHED, RELATED and INVALID
- NEW - tells us that the packet is the first packet that we see from a specific connection
- ESTABLISHED - next packets from the same connection

Firewall - iptables - connection state

- Packets can be related to tracked connections in four different so called states: NEW, ESTABLISHED, RELATED and INVALID
- NEW - tells us that the packet is the first packet that we see from a specific connection
- ESTABLISHED - next packets from the same connection
- RELATED - new connection created by already ESTABLISHED connection

Firewall - iptables - connection state

- Packets can be related to tracked connections in four different so called states: NEW, ESTABLISHED, RELATED and INVALID
- NEW - tells us that the packet is the first packet that we see from a specific connection
- ESTABLISHED - next packets from the same connection
- RELATED - new connection created by already ESTABLISHED connection
- INVALID - packet not identified

Firewall - iptables - NAT

- There are two types of NAT: Source NAT (SNAT) and Destination NAT (DNAT)

  Change the source address of a packet to 1.2.3.4

Firewall - iptables - NAT

- There are two types of NAT: Source NAT (SNAT) and Destination NAT (DNAT)

  Change the source address of a packet to 1.2.3.4

  root@Harvey:# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4

Firewall - iptables - NAT

- There are two types of NAT: Source NAT (SNAT) and Destination NAT (DNAT)

  Change the source address of a packet to 1.2.3.4

  root@Harvey:# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4

  Change the destination address of a packet to 1.2.3.4

Firewall - iptables - NAT

- There are two types of NAT: Source NAT (SNAT) and Destination NAT (DNAT)

  Change the source address of a packet to 1.2.3.4

  root@Harvey:# iptables -t nat -A POSTROUTING -o eth0 -j SNAT --to 1.2.3.4

  Change the destination address of a packet to 1.2.3.4

  root@Harvey:# iptables -t nat -A PREROUTING -i eth1 -j DNAT --to 1.2.3.4

Firewall - iptables - NAPT

- Forwarding ports - process of forwarding packets, which are being received on a specific port.

Firewall - iptables - NAPT

- Forwarding ports - process of forwarding packets, which are being received on a specific port.

  We want to forward packets from interface eth0 port 80 to 192.168.1.200 to port 8080

Firewall - iptables - NAPT

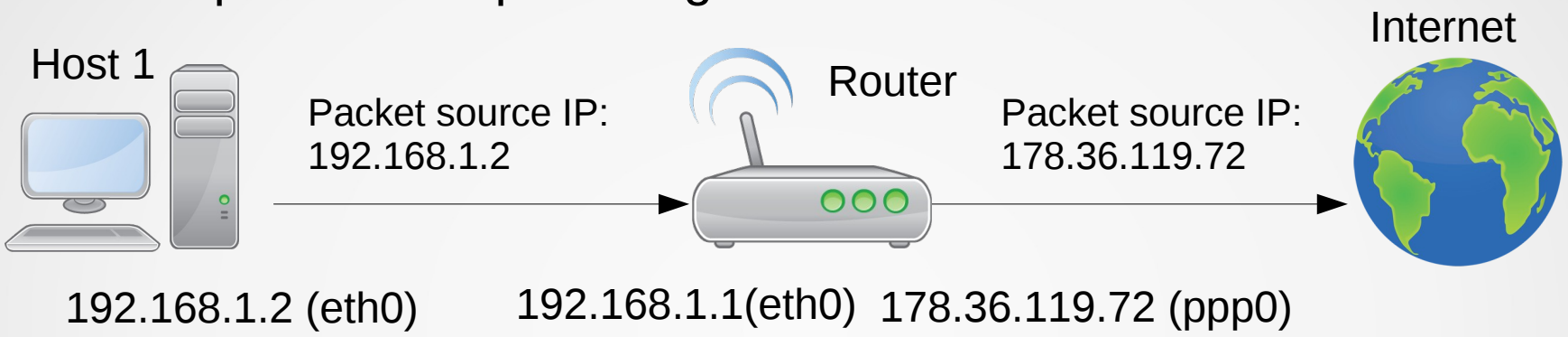
- Forwarding ports - process of forwarding packets, which are being received on a specific port.

  We want to forward packets from interface eth0 port 80 to 192.168.1.200 to port 8080

  iptables -t nat -A PREROUTING -i eth0 --dport 80 -j DNAT --to-destination 192.168.1.200:8080

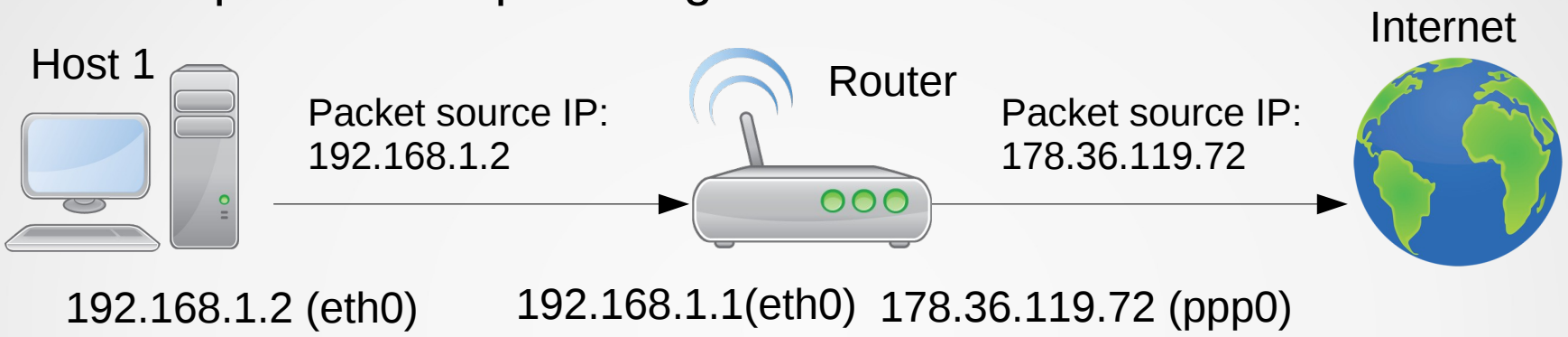
Firewall - iptables - more examples

- 1st Example: IP Masquerading



Firewall - iptables - more examples

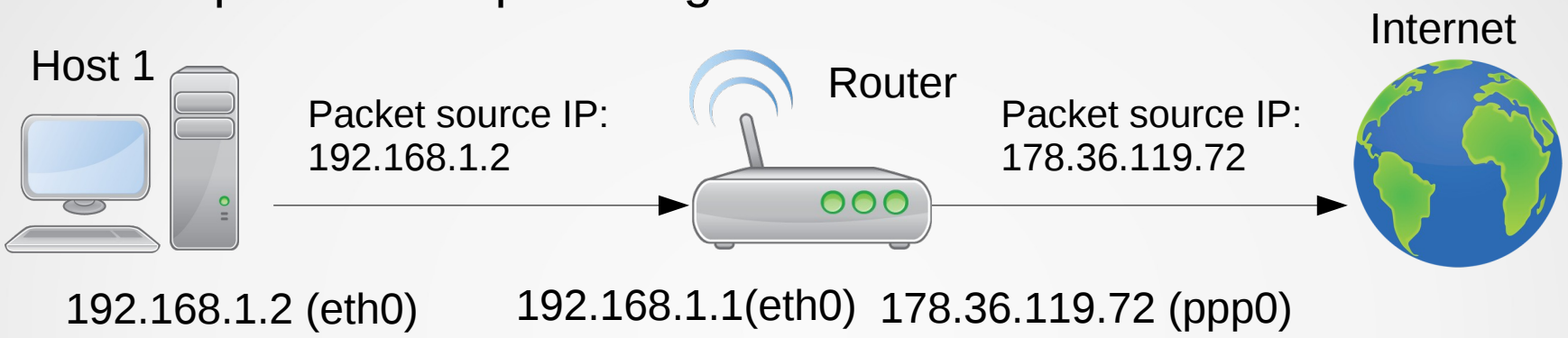
- 1st Example: IP Masquerading




```
root@Router:# echo 1 > /proc/sys/net/ipv4/ip_forward
root@Router:# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE
```

Firewall - iptables - more examples

- 1st Example: IP Masquerading



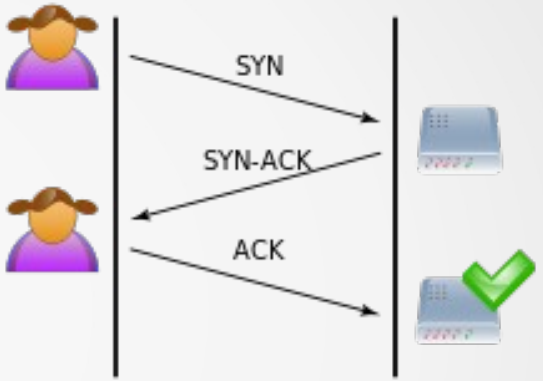

```
root@Router:# echo 1 > /proc/sys/net/ipv4/ip_forward
root@Router:# iptables -t nat -A POSTROUTING -o ppp0 -j MASQUERADE

root@Host1:# ping 8.8.8.8
PING 8.8.8.8 (8.8.8.8) 56(84) bytes of data.
64 bytes from 8.8.8.8: icmp_seq=1 ttl=46 time=34.5ms
64 bytes from 8.8.8.8: icmp_seq=2 ttl=46 time=34.0ms
```

Firewall - iptables - more examples

- 2nd Example: Blocking syn-flood attacks

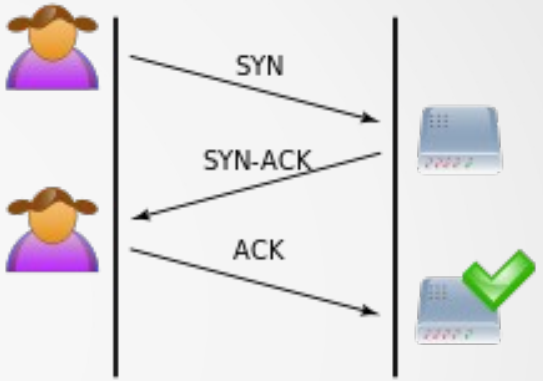
Firewall - iptables - more examples

- 2nd Example: Blocking syn-flood attacks
- In TCP SYN attack, target's computer is being flooded with TCP SYN requests



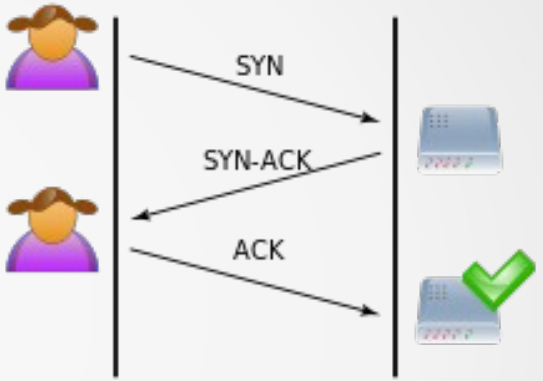
Firewall - iptables - more examples

- 2nd Example: Blocking syn-flood attacks
- In TCP SYN attack, target's computer is being flooded with TCP SYN requests
- To resolve this problem we will limit incoming SYN requests



Firewall - iptables - more examples

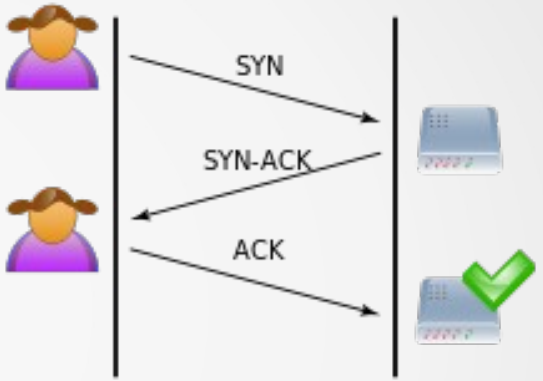
- 2nd Example: Blocking syn-flood attacks
- In TCP SYN attack, target's computer is being flooded with TCP SYN requests
- To resolve this problem we will limit incoming SYN requests




```
root@Computer:# iptables -N syn_flood
root@Computer:# iptables -I INPUT 1 -p tcp -j syn_flood
```

Firewall - iptables - more examples

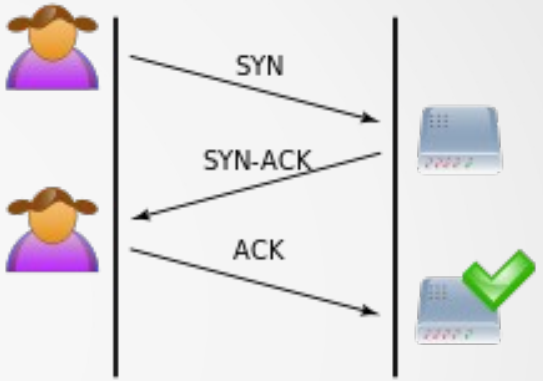
- 2nd Example: Blocking syn-flood attacks
- In TCP SYN attack, target's computer is being flooded with TCP SYN requests
- To resolve this problem we will limit incoming SYN requests




```
root@Computer:# iptables -N syn_flood
root@Computer:# iptables -I INPUT 1 -p tcp -j syn_flood
root@Computer:# iptables -A syn_flood --syn -m limit --limit 1/s --limit-burst 3 -j RETURN
root@Computer:# iptables -A syn_flood -j DROP
root@Computer:# iptables -L
```

Firewall - iptables - more examples

- 2nd Example: Blocking syn-flood attacks
- In TCP SYN attack, target's computer is being flooded with TCP SYN requests
- To resolve this problem we will limit incoming SYN requests




```
root@Computer:# iptables -N syn_flood
root@Computer:# iptables -I INPUT 1 -p tcp -j syn_flood
root@Computer:# iptables -A syn_flood --syn -m limit --limit 1/s --limit-burst 3 -j RETURN
root@Computer:# iptables -A syn_flood -j DROP
root@Computer:# iptables -L
```

```
Chain INPUT (policy ACCEPT)
target     prot opt source               destination
syn_flood  tcp  --  anywhere             anywhere
ufw-before-logging-input  all  --  anywhere             anywhere

Chain syn_flood (1 references)
target     prot opt source               destination
RETURN     all  --  anywhere             anywhere             limit: avg 1/sec burst 3
DROP       all  --  anywhere             anywhere
```

Firewall - xtables-addons

- *xtables-addons* is a package containing extensions for iptables

Firewall - xtables-addons

- *xtables-addons* is a package containing extensions for iptables
- Ubuntu: *apt-get install xtables-addons-common*

Firewall - xtables-addons

- *xtables-addons* is a package containing extensions for iptables
- Ubuntu: *apt-get install xtables-addons-common*
- Example extensions:
  - geoip
    - allows matching on countries

Firewall - xtables-addons

- *xtables-addons* is a package containing extensions for iptables
- Ubuntu: *apt-get install xtables-addons-common*
- Example extensions:
  - geoip
    - allows matching on countries
  - tarpit
    - extends tcp matching functionality

Firewall - xtables-addons

- *xtables-addons* is a package containing extensions for iptables
- Ubuntu: *apt-get install xtables-addons-common*
- Example extensions:
  - geoip
    - allows matching on countries
  - tarpit
    - extends tcp matching functionality
  - account
    - provides statistics for packets (counting packets etc)