Network Administration

Alessandro Barenghi

Dipartimento di Elettronica, Informazione e Bioingegneria
Politecnico di Milano

barenghi - at - elet.polimi.it

April 9, 2013

Summary of the talk

Teaching you how to deal with networking

- Managing your own host:
- Packet Filtering (a.k.a. Firewalling)
- Source and Destination Network Address Translation

Caveats and prerequisites

What you should already know

- This talk is intended for all audiences
- However, having an understanding of how a TCP/IPv4/Ethernet based network works will help a lot
- For those who do, you can play `hunt` during the next slides
- For those who do not have a clue, I'll do my best to sum up the key concepts in the next slide[a] but you *should* dig deeper

[a] Jon Postel forgive me, for I know what I'll be doing...

TCP/IPv4/Ethernet based networks, *brutalized*

The ISO/OSI Stack

- Network communications are managed by a set of protocols (ways to communicate among hosts)
- Protocols are organized as a stack: manage everything from the physical level, up to the bytes you get from the network
- Protocols on the bottom of the stack do their best to deliver the data produced by higher levels
- Higher levels are closer to the user, lower are close to the Iron
- Think of the whole stack as all the things involved in the postal service: streets, postmen, buildings, apartments

TCP/IPv4/Ethernet based networks, *brutalized*

Level 2: Ethernet

- The Ethernet layer is the 2nd layer of our stack, handles a couple of things above the cables
- It is the equivalent of the common streets, crossroads and semaphores:
  - Allows entities to carry something from a place to another, when incapsulated in cars
  - Handles collisions when they do happen (in the case of Ethernet, it simply sends again the information)
- A place (host) is identified by a 6-byte Ethernet address
  - Colloquially, it is also called MAC address as the 2nd layer of the stack is the Medium Access Control

TCP/IPv4/Ethernet based networks, *brutalized*

Layer 3: IPv4

- The IPv4 layer acts pretty much as postal lorries, carrying around our packets (moving on the streets)
- The IPv4 layer provides a best-effort delivery of our data to a place in the network
- An IPv4 packet should thus be routed around the network until it reaches the destination (or dies out of boredom)
- The IPv4 addresses are 4-byte wide and usually written as 4 dot-separated decimal digits, `a.b.c.d`, e.g. 216.34.181.45

TCP/IP based networks, *brutalized*

Layer 4: TCP

- Since postal lorries get lost every now and then, we send our data with return receipt letters :)
- TCP is the layer of the stack which handles the return receipts, and sends them again in case they fail to get there
- Additionally, the TCP layer allows multiple connections from the same IP address to another
- The connections are made on different ports , identified by a 16 bit number (0-65535)

Network Administration

Local?

- The first step to become network administrator is to manage our host :)
- Local host management requires to configure properly:
  - Check that Ethernet (level 2) is ok: pave the road
  - IPv4 addresses (level 3): select your building/block
  - Configure the Routing Tables (level 3): know the streets
  - Firewalling (Level 3-4): meet new people and lock doors
- I'll try to be as distribution agnostic as possible, only permanent network configurations will differ
- Still, you'll have the knowledge required to cope with that

Network configuration

Interfaces

- The networking architechture of Linux is based on interfaces
- Interfaces are the points of contact with the rest of the world
- Interfaces can be of various types, among which:
  - Physical pieces of Iron (e.g. network card)
  - An endpoint of a virtual channel (Tunnels and VPNs)
  - A virtual confluence of two or more interfaces (bridges)
- For all the purposes of this talk, we won't care how they send the data, as long as they do

The iproute2 suite

One tool to bind them all

- Management of the network levels 2/3 is done with the `iproute2` suite [a]
- The commands all share the same structure: `ip [options] object command`
- `ip link` and `ip neigh` manage the MAC level (physical interconnections addresses)
- `ip addr` and `ip route` manage Level 3 (IP addresses)
- Level 4 traffic control is demanded to the `tc` tool and the `NetFilter/IpTables` suite

[a] Management of anything below the interface is usually better done with a proper soldering iron

iproute2 - Ethernet control (Level 2)

Bringing up the communications/ Paving the road

- Interface management is done using the `link` keyword
- `ip link set <iface> [up|down]` will enable/disable an interface
- `ip link show` will list all the interfaces and show their MAC addresses (typical format : XX:XX:XX:XX:XX:XX)
- `ip link set <iface> address <MAC address>` changes the interface current MAC address with something else
- `ip link set <iface> arp [on|off]` toggles the ARP protocol, in case you do not want it

iproute2 - IPv4 Address control (Level 3)

Setting addresses

- Once we have a working interface, we can assign one (or more) addresses to it via the `addr` keyword
- `ip addr add <IP address>/<netmask length> dev <interface>` will add an address to an interface
- `ip addr del <IP address>/<netmask length> dev <interface>` removes an address from the interface
- `ip addr flush <iface>` will wipe all the addresses from the interface
- `ip address show` will simply list the ip addresses assigned to the interfaces

iproute2 - Dynamic Addresses

Getting an address

- IP addresses can also be assigned dynamically if a proper DHCP server is connected to the interface
- Useful when you have a large number of hosts continuously connecting and disconnecting (say, if you're an ISP)
- The DHCP server will, upon query, lend an address to the client for a limited amount of time (lease time)
- The most common DHCP clients available are `dhcpcd`,`dhclient` and `pump`
- In all cases to obtain an address for an interface, simply go with `<client_name> <iface>`
- This corresponds to the *auto* setting of all your favourite distribution tools

Subnets

Who lives in my block?

- Once we have our IPv4 address, we need to know where the rest of the people are located
  - That is, given a destination IPv4 address, which interface should we use send the packet to its destination
- To make this easier, we employ subnets (=groups of IPs)
- The common way to indicate a subnet is together with an IPv4 address belonging to it, e.g., 192.168.0.1/255.255.255.0
- An IPv4 is in the subnet if `IP&Subnet Mask`=0
- In the example, the addresses 192.168.0.[0-255] are in the subnet

Network Layer

Routing

- Each of our interfaces has an IP and subnet associated
- If we want to talk to someone in the subnets, we already know where to send the data
- We need to manage when we do not know where someone is: we need a default gateway (= an host which will deliver the data for us)
- Note: the default gateway must be on one of our subnets (or we won't know how to contact it)
- We will thus build a routing table, where to look when we want to send stuff

Network Layer

Routing

- Adding a route is as simple as `ip route add <address>/<mask length> via <address>`
- You can enforce the packets down a specific interface by adding `dev <interface>` at the end
- To remove a route simply use `ip route del <address>/<mask length> via <address>`
- The default route is specified as the destination for the 0.0.0.0/0.0.0.0 subnet, as it will match anything
- If two routes match the destination of the packet, the one with the longest subnet is matched

Host Network Status

The Socket Stats tool `ss`

- Invoking the tool without parameters lists all the communications (sockets) open on the host
- By default the *known ports* are listed with the service name instead of their port number

Packet filtering

Where?

The (main) firewall should be the single point of contact between the secure and insecure zone

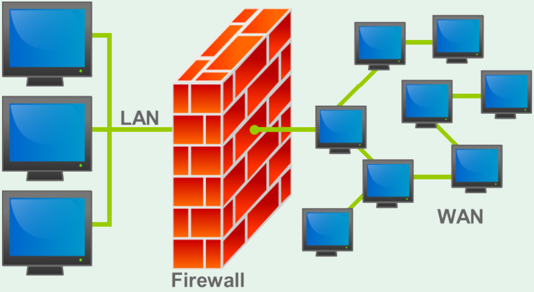


Figure : Firewall Placement

Packet filtering

Why firewalling?

- Avoiding unauthorized connections regardless of the availability of a server
- Packet sanitization (checksum check) can be performed during filtering
- Checking correctness of TCP and higher protocols behaviours
- Network and Port Address translation strategies can be applied

Firewalling with Linux

The Netfilter/Iptables suite

- The Linux kernel provides all the facilities to perform proper packet filtering
- The stateful packet filtering infrastructure is known as NetFilter
- The infrastructure is made up of 5 hooks on the main paths where the packets are passing
- It is possible to define trigger-based rules matching a specific class of packets
- They can be either allowed to pass or redirected/discarded/mangled
- In order to interact with the packet filtering infrastructures we will use the `iptables` command

Structure

Overview

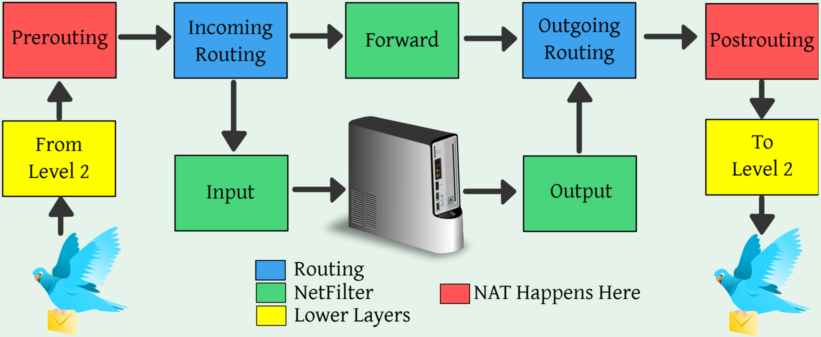


Figure : NetFilter Structure

Netfilter tables

Overview

- Every Netfilter table is made up of a list of rules, checked in order
- If no rule matches the packet, the default action, i.e. the chain policy is adopted
- Up to four tables containing chains are present (filter,nat, mangle and raw) for each Netfilter hook
- It is possible to create custom tables of rules in order to avoid the crowding of the default chain
- There is no possibility to add hook structures by default [a]

---

[a] obviously, you can write an extra kernel module, but that's not exactly "by default"

Hook policies

Setting the defaults

- Every builtin chain has a default policy, i.e. a default action to be performed on the packet
  - `ACCEPT`: the packet flows through the hook, towards its destination
  - `QUEUE`: the packet is sent to the userspace via Netlink for examination
  - `DROP`: the packet is discarded and treated as it never existed
- A hook policy can be set up with `iptables -P <chain> <policy>`
- The default policy, with which the kernel boots Netfilter is `ACCEPT` for all the base chains

Hook policies

Reasonable policies

- Reasonable policies usually are :
  - PRE/POSTROUTING: set to ACCEPT, these chains are not meant for dropping
  - INPUT: set to DROP, whitelist is better than blacklist
  - FORWARD: set to DROP, "Thou shall not pass" is a reasonable default for the same reasons
  - OUTPUT: set to ACCEPT, although particularly restrictive policies may need a DROP

Rules - management

Rule structure

- The Netfilter behaviour is modified via the `iptables` command
- A rule is composed of two parts, the match and the target
- The match specifies the conditions regarding the packet which will trigger the rule
- The target specifies the fate of the packet
- For basically all match specifications , prepending a `!` mark inverts the match

Rules - management

Targets

- Possible targets (with extensions) for a rule are :
  - `ACCEPT/DROP` : behave exactly as the policies
  - `REJECT`: The packet is dropped but, if allowed by the protocol, the sender is notified of the rejection
  - `LOG`: A line in the kernel log is written, and the check on the chain of rules goes on
  - `MIRROR`: Swaps source and destination address and immediately sends the packets without passing via the other chains
  - `RATEEST`: adds this packet to the statistic of a rate estimator, then the chain checks go on

Rules - management

Rule management

- The generic iptables command is structured as : `iptables [-t table] <action> <rule>`
- Possible actions are :
  - `-A <chain>` : appends a rule at the end of the chain
  - `-D <chain>` : deletes the specific rule (the number of the rule may be indicated instead)
  - `-I <chain> <num>`: inserts the rule as the n-th
  - `-R <chain> <num>`: replaces the n-th rule
  - `-L`: lists all the rules of a chain
  - `-F`: flushes a chain (but does not reset the policy to `ACCEPT`)

Rules - 1

Matching interfaces

- The first and most simple match for a packet is to decide an action depending on the interface it was received on
- The inbound/outbound interface matches are specified via the `-i <iface>`/`-o <iface>` option
- The `-i`/`-o` options are limited to some chains, namely:
  - `-i` can only be used in `INPUT`, `FORWARD` and `PREROUTING`
  - `-i` can only be used in `OUTPUT`, `FORWARD` and `POSTROUTING`
- The most common use of this match is to differentiate the reasonably trusted zone of the network (LAN side) from the really untrusted side (WAN side)

Rules - 1

Matching interfaces - 2

- A special case for interface matching is the loopback interface `lo`
- This interface should never be filtered, lest a couple of applications *will* misbehave
- Accepting all packets with destination address equal to 127.0.0.1 is not equivalent to accepting `lo` (See RFC3330)
- Accepting all packets with destination address equal to 127.0.0.0/8 is not equivalent to accepting `lo` either (packets directed to an address you own are routed to `lo` when you self connect)

Rules - 2

Matching Addresses and ports

- The most common match is the one checking either the source `-s` or the destination `-d` address
- It is possible to specify the mask as the number of contiguous ones `/n` or explicitly `/a.b.c.d`
- If the rule does not specify any mask, the default is /32, i.e. host only
- Also non contiguous masks are usable: e.g. 255.255.255.249 (0xFFFFFFF9) matches all the odd hosts up to .7
- Employing non contiguous masks may help in reducing the number of rules

Rules - 2

Matching Addresses

- The most common match is the one checking either the source `-s` or the destination `-d` address
- It is possible to specify the mask as the number of contiguous ones `/n` or explicitly `/a.b.c.d`
- If the rule does not specify any mask, the default is /32, i.e. host only
- Also non contiguous masks are usable: e.g. 255.255.255.249 (0xFFFFFFF9) matches all the odd hosts up to .7
- Employing non contiguous masks may help in reducing the number of rules

Configuration Management

Saving and Restoring

- The `iptables` utility updates a rule at a time via Netlink
- In case multiple rule changes should be performed atomically it is not a good idea to call it a volley of times
- The `iptables-apply` is able to insert atomically the changes in the Netfilter tables
- The `iptables-save` and `iptables-restore` command provide a way of dumping and restoring a full ruleset at once
- There is also an `iptables-xml` utility which converts a ruleset in XML for whatever purposes it may have

Address Translation

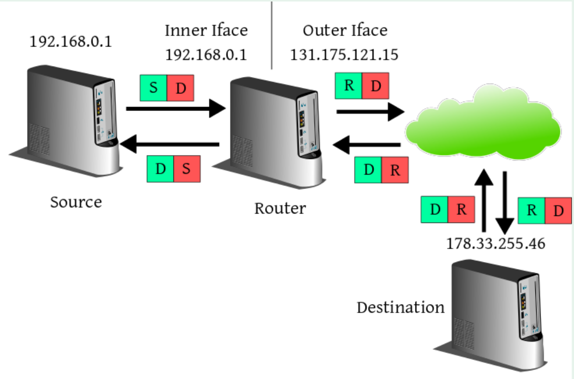
Source NAT

- Although it raises some issues, it may be needed to mask a number of hosts under a single one
- Common when a LAN needs to access a public network but only one public IP address has been bought from IANA or the ISP
- Useful to “concentrate” accesses behind a single IP
- The best candidate to perform the packet mangling is the router

Address Translation

SNAT

The general structure of a source network address translation based architecture



Address Translation

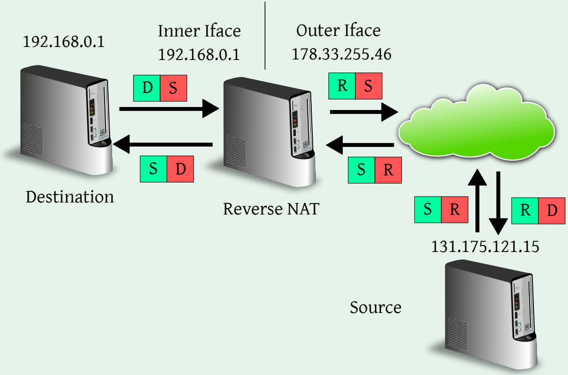
Destination NAT

- Symmetrically, it may be helpful to split the network load managed by a server
- This can be done through dynamically modifying the destination of the communication
- The operation must be performed by the alleged target of the communications
- Bonus: it allows to replace machines behind the DNAT without interrupting a service

Address Translation

DNAT

The general structure of a destination network address translation based architecture



Source NAT

Overview

- Source NAT is performed in the `POSTROUTING` hook, when the packet is about to leave
- The corresponding translation for the returning packet is automatically managed
- A simple `-t nat -A POSTROUTING -j SNAT --to <address>` rule sets all the packets matching it to be masked
- The output interface specifying option `-o` and comes in handy to specify which connection to mask
- The special target `-j MASQUERADE` instructs Netfilter to choose automatically the outgoing address according to the egress interface

Destination NAT

Overview

- Destination NAT is performed (symmetrically) in the `PREROUTING` hook, before anything is done to the packets
- The bidirectional communication of an established connection is also automatically managed
- The `-t nat -A PREROUTING -j DNAT --to-destination <address>` rule indicates the address where the packet should be redirected
- The input interface specifying option `-i` allows rough balancing in multi-interface routers
- Obviously, no automatic destination selection can be performed here