

Proxy e Firewall

Stefano Sasso
stefano @ faberlibertatis.org

A.P.S. Faber Libertatis
Serate di formazione tecnica interna

f a b e r  l i b e r t a t i s

Introduzione...

- Lo stack TCP/IP
- Indirizzo MAC e indirizzo IP
- Principali dispositivi di rete
- Indirizzo IP e maschera di rete
- Routing
- Masquerading (NAT)

Lo stack TCP/IP

Tutte le informazioni che transitano in internet sono basate su una pila di protocolli; tale pila a volte è chiamata TCP/IP (stack tcp/ip) in funzione dei due principali protocolli in essa presenti: il protocollo TCP e il protocollo IP.

Questi protocolli si possono dividere in 5 macro-categorie (comunemente detti livelli), a seconda del loro scopo:

livello 1: Livello Fisico (es: doppino, fibra ottica, ...)

livello 2: Livello di Collegamento (es: ethernet, wifi, ppp, token ring, ...)

livello 3: Livello di Networking (es: IPv4, IPv6, ICMP, BGP, ...)

livello 4: Livello di Trasporto (es: TCP, UDP, ...)

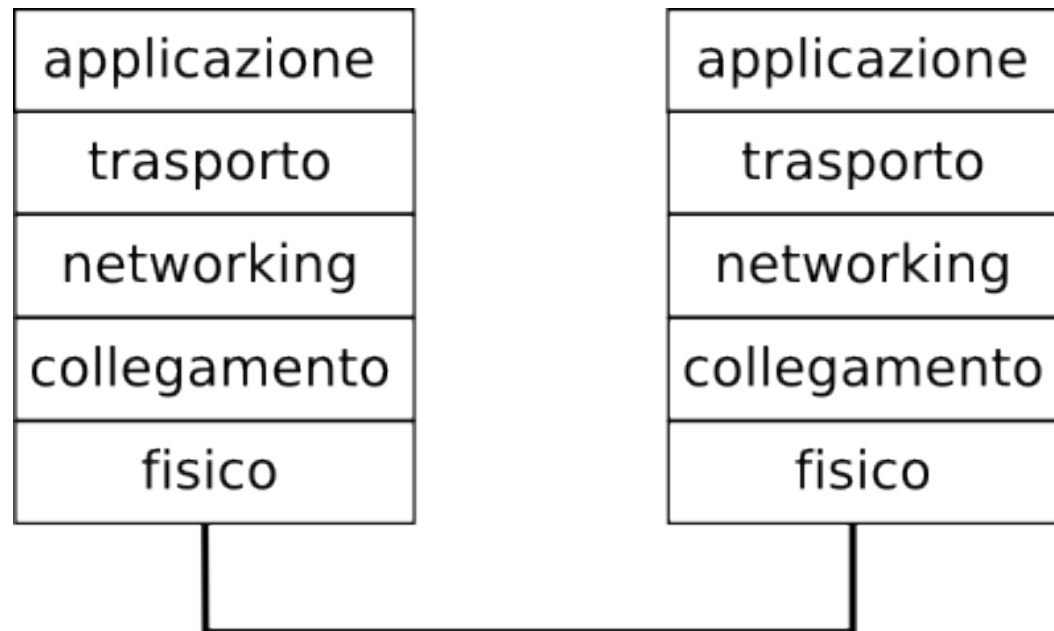
livello 5: Livello Applicazioni (es: HTTP, SMTP, SSH, FTP, DNS)

una nota:

In realtà questa pila di protocolli ricade sotto l'astratta pila ISO/OSI, formata da 7 livelli; per questo a volte il **livello di Applicazione (5)** viene detto livello 7.

Il percorso dei pacchetti:

Partendo dalla sorgente, i pacchetti percorreranno i livelli 5, 4, 3, 2, 1; mentre a destinazione percorreranno i livelli 1, 2, 3, 4, 5. Tutto questo se non c'è niente di mezzo.



indirizzo MAC e IP

L'indirizzo MAC si piazza al livello 2, mentre l'indirizzo IP al livello 3

applicazione
trasporto
networking
collegamento
fisico

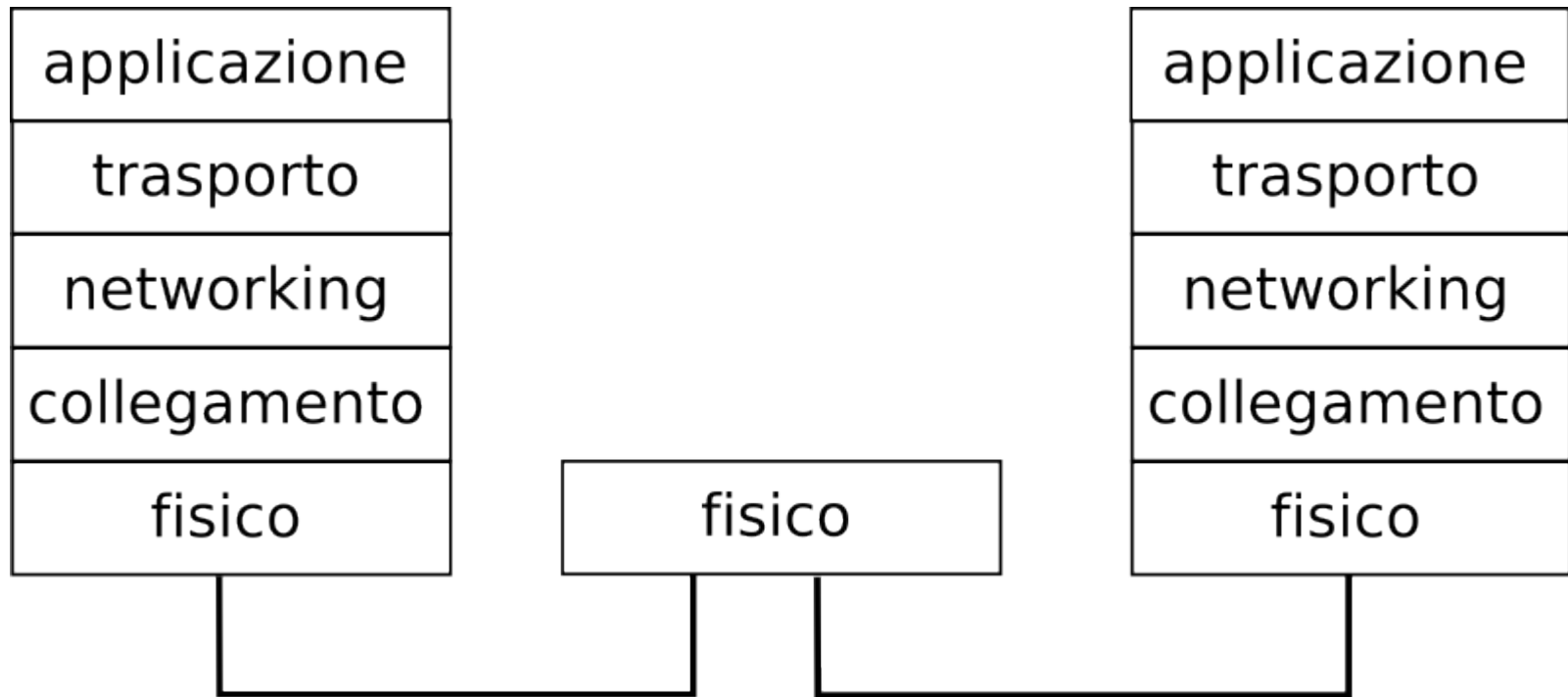
IP: 192.168.176.193

MAC: 00:AE:43:12:A6:67

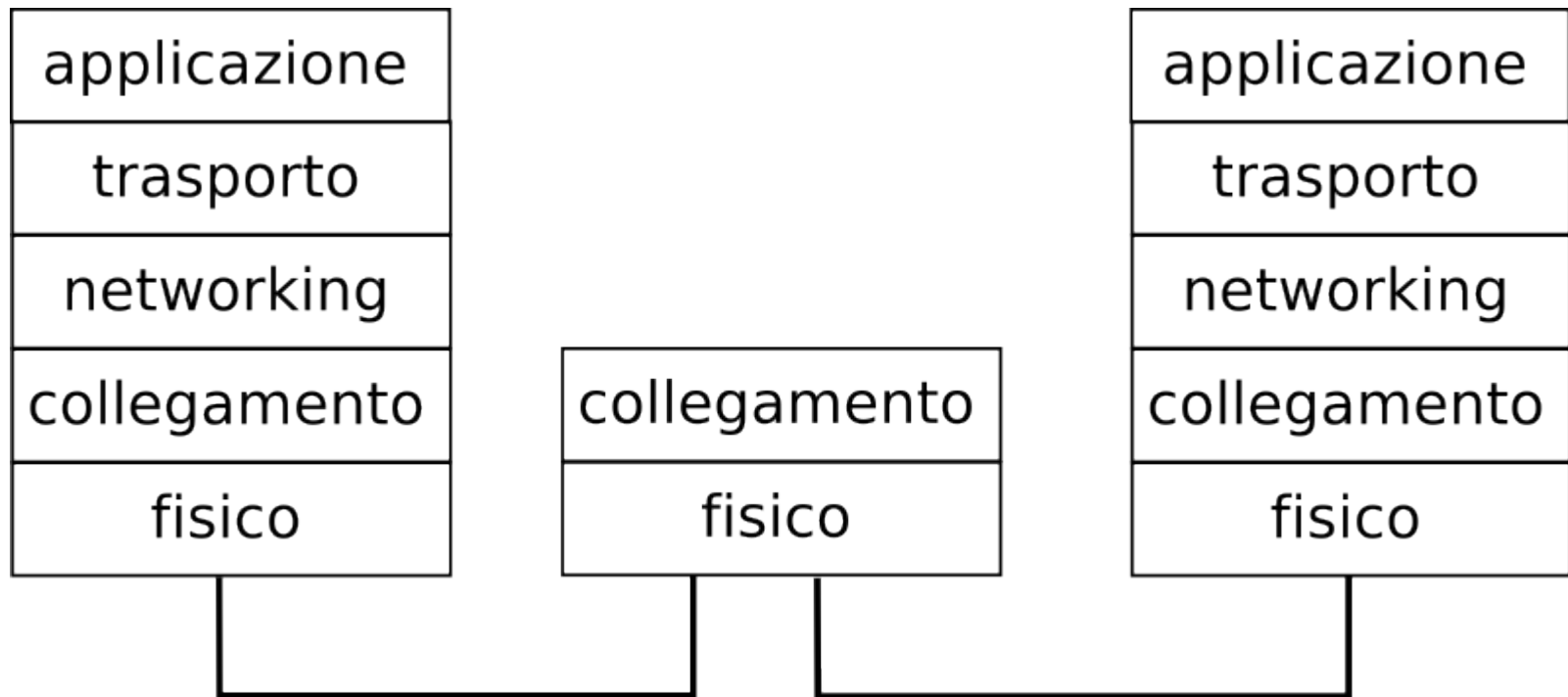
I principali dispositivi di rete

- Hub (repeater)
- Switch / Bridge
- Router
- ...
- Proxy

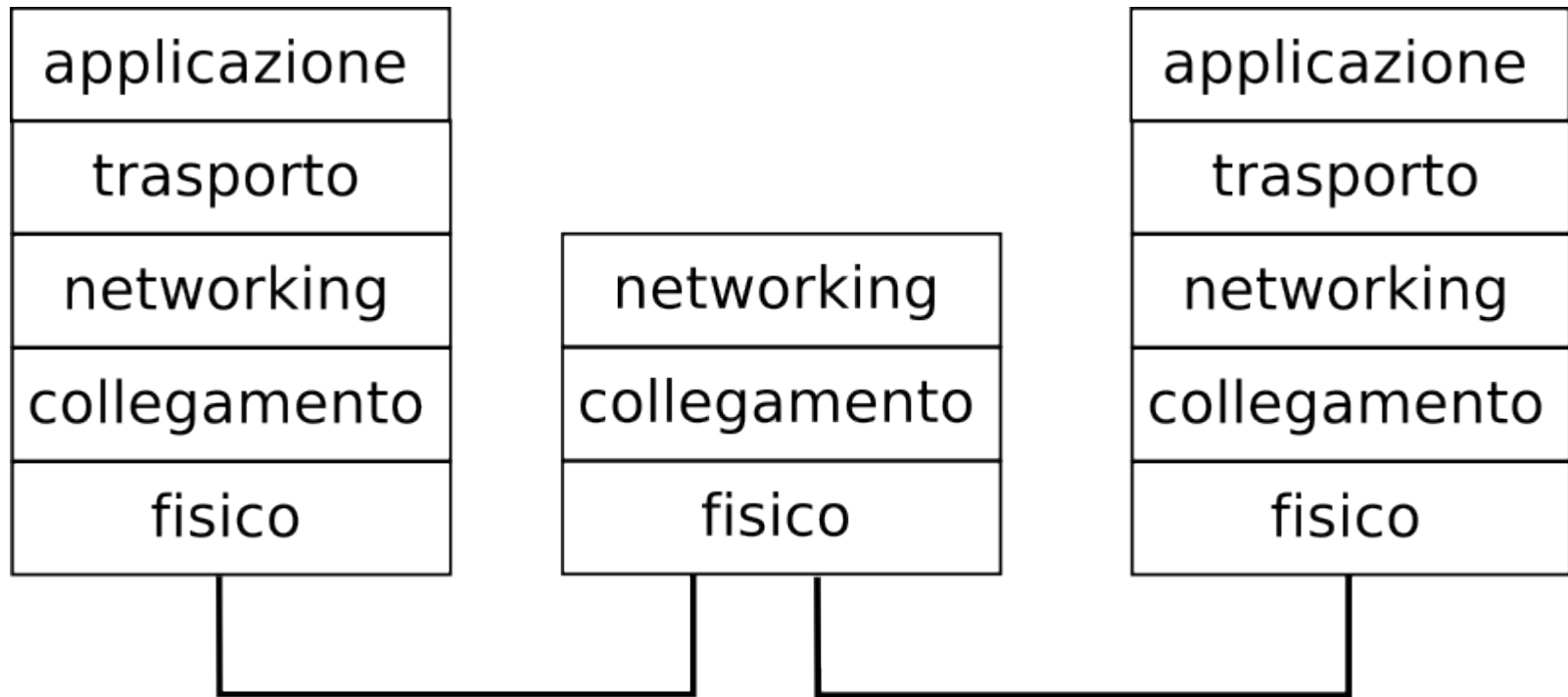
dispositivi: hub



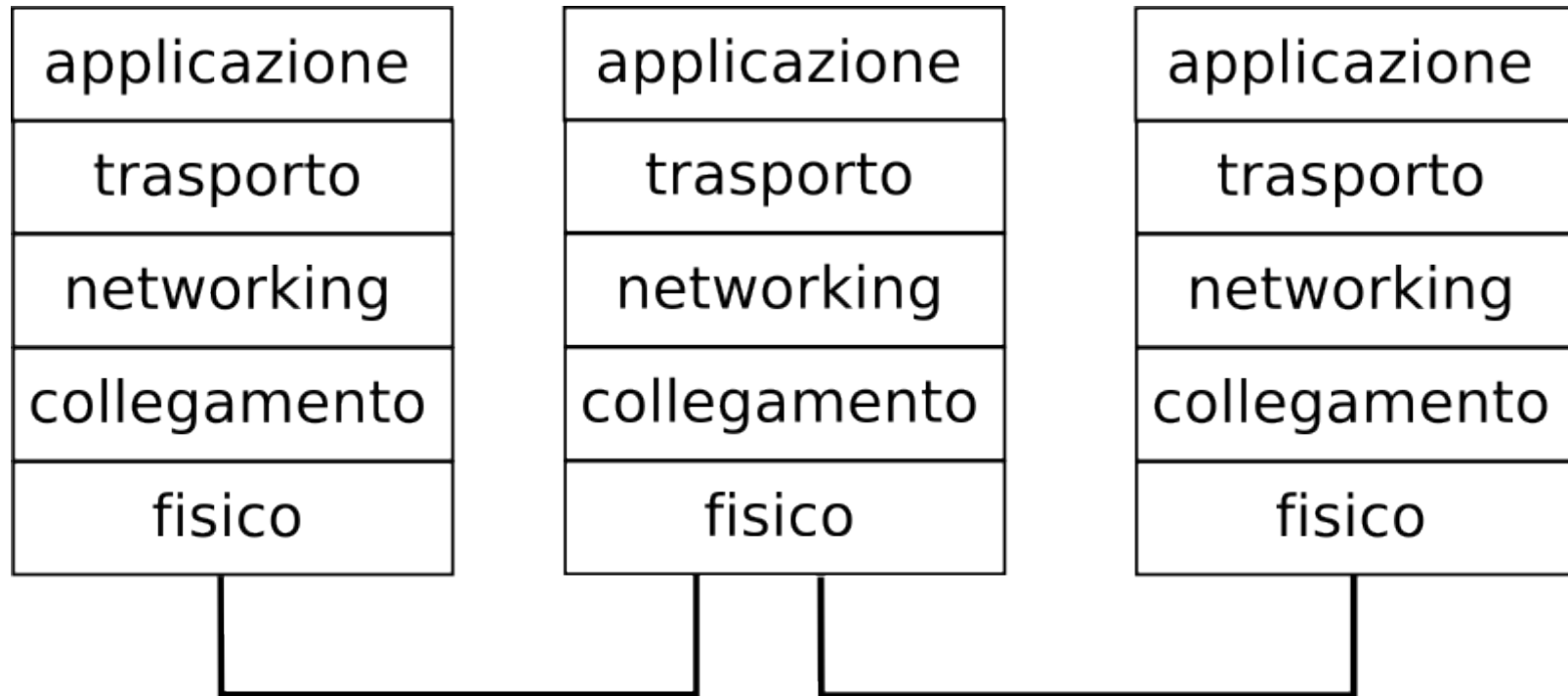
dispositivi: switch



dispositivi: router



dispositivi: proxy



L'indirizzo IP

L'indirizzo ip è un "numero" che identifica univocamente un computer in una singola rete; in particolare, ad ogni interfaccia fisica collegata ad una rete può venire assegnato uno o più indirizzi ip.

Un indirizzo IPv4 è composto da 4 otteti separati da punto, ovvero da 4 numeri decimali 0-255 separati da un punto.

esempio: 192.168.1.1 194.177.95.205
10.0.15.193

Maschera di rete

Una maschera di rete associata ad un'indirizzo ip permette di definire quali sono gli host direttamente raggiungibili senza passare per un router.

Ad esempio, l'indirizzo ip 192.168.32.97 con maschera di rete 255.255.255.0, tramite operazione AND bit a bit, definisce che, ad esempio, l'host 192.168.32.130 è direttamente raggiungibile. (è nella stessa sottorete)

192 . 168 . 032 . 097	AND	192 . 168 . 032 . 130	AND
255 . 255 . 255 . 000	=	255 . 255 . 255 . 000	=
192 . 168 . 032 . 000		192 . 168 . 032 . 000	

ancora maschere...

Il primo e l'ultimo indirizzo di una sottorete (nel nostro esempio, 192.168.32.0 e 192.168.32.255) sono riservati (non possono essere assegnati a nessun computer) e vengono chiamati, rispettivamente, **indirizzo di rete** e **indirizzo di broadcast**.

Le maschere di rete sono delle forme binarie simili all'indirizzo ip, ma non possono avere 0 alternati ad 1;

valida: 11111111 . 11111111 . 11111111 . 11100000

non valida: 11111111 . 11111111 . 11111111 . 10111010

è possibile indicare le maschere di rete in due notazioni; la prima, in maniera simile ad un indirizzo ip:

es: 255.255.255.0

la seconda, facendo seguire l'indirizzo ip da un numero decimale rappresentante il numero di *uni* della maschera stessa:

es: 192.168.1.15/24

Alcuni intervalli di indirizzi ip sono stati riservati a delle reti private (LAN), ma non possono transitare in internet. (in internet possono transitare solo ip pubblici).

questi intervalli sono:

10.0.0.0-10.255.255.255 (o, più precisamente, **10.0.0.0/8**)

172.16.0.0-172.31.255.255 (**172.16.0.0/12**)

192.168.0.0-192.168.255.255
(**192.168.0.0/16**)

sottoreti

si noti che, usando opportune maschere di rete, è possibile segmentare qualsiasi rete in reti più piccole;

un uso molto frequente è la segmentazione di 192.168.0.0/16 in sottoreti 192.168.X.0/24, da 256 indirizzi ciascuna (0-255, di cui 254 utilizzabili).

altri esempi di segmentazioni sono:

192.168.10.0/30 62.100.95.8/29 172.17.0.0/16

etc...

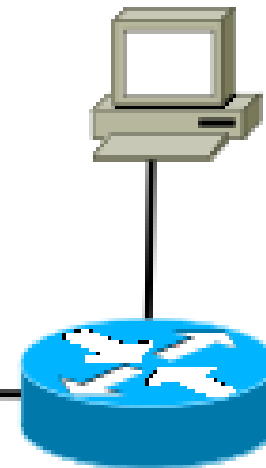
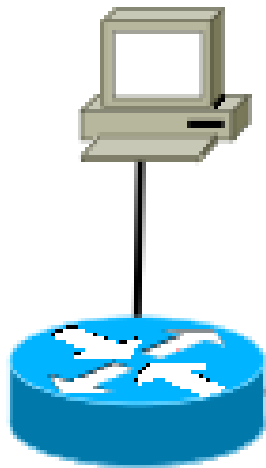
Il routing, o instradamento, viene effettuato a livello 3; nel caso tipico di IP, i router conservano delle tabelle di instradamento in cui sono definiti i modi per raggiungere determinate destinazioni. Le tabelle di instradamento possono essere popolate con una combinazione di diversi modi:

- ✓ reti fisicamente connesse
- ✓ routing statico
- ✓ routing dinamico

routing

IP: 192.168.1.1/24
Tabella di routing:
192.168.1.0/24 dev eth0
default via 192.168.1.254

IP: 192.168.2.1/24
Tabella di routing:
192.168.2.0/24 dev eth0
default via 192.168.2.254



IP: 192.168.1.254/24
IP: 192.168.250.1/30
rotte:
192.168.1.0/24 dev eth0
192.168.250.0/30 dev eth1
192.168.2.0/24 via 192.168.250.2

IP: 192.168.2.254/24
IP: 192.168.250.2/30
rotte:
192.168.2.0/24 dev eth0
192.168.250.0/30 dev eth1
192.168.1.0/24 via 192.168.250.1

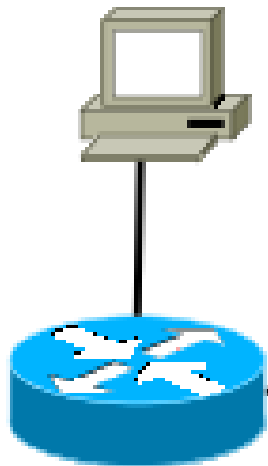
Il masquerading serve per mascherare una serie di indirizzi ip in un unico indirizzo.

è da usarsi solo in due casi:

1. mascheramento rete privata in un ip pubblico verso internet (ovvio!!!)
2. qualora non si possa agire nelle tabelle di routing (un po' meno banale, vediamo insieme perchè)

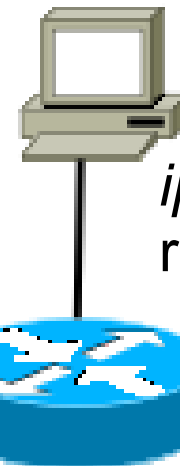
masquerading (caso 2)

IP: 192.168.1.1/24
Tabella di routing:
192.168.1.0/24 dev eth0
default via 192.168.1.254



IP: 192.168.1.254/24
IP: 192.168.250.1/30
rotte:
192.168.1.0/24 dev eth0
192.168.250.0/30 dev eth1
192.168.2.0/24 via 192.168.250.2

IP: 192.168.2.1/24
Tabella di routing:
192.168.2.0/24 dev eth0
default via 192.168.2.254

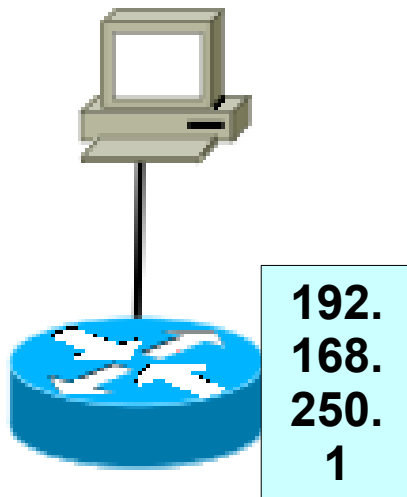


ipotesi: Questo router non è nostro

IP: 192.168.2.254/24
IP: 192.168.250.2/30
rotte:
192.168.2.0/24 dev eth0
192.168.250.0/30 dev eth1
~~192.168.1.0/24 via 192.168.250.1~~

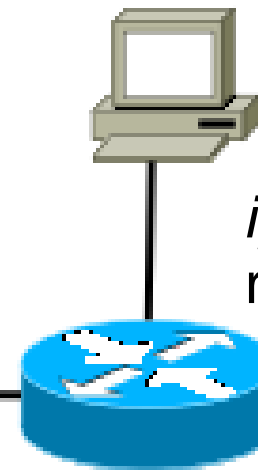
masquerading (caso 2)

IP: 192.168.1.1/24
Tabella di routing:
192.168.1.0/24 dev eth0
default via 192.168.1.254



IP: 192.168.1.254/24
IP: 192.168.250.1/30
rotte:
192.168.1.0/24 dev eth0
192.168.250.0/30 dev eth1
192.168.2.0/24 via 192.168.250.2

IP: 192.168.2.1/24
Tabella di routing:
192.168.2.0/24 dev eth0
default via 192.168.2.254



ipotesi: Questo router non è nostro

IP: 192.168.2.254/24
IP: 192.168.250.2/30
rotte:
192.168.2.0/24 dev eth0
192.168.250.0/30 dev eth1
~~192.168.1.0/24 via 192.168.250.1~~

Per abilitare il routing tra più interfacce di rete è necessario dare

```
echo 1 > /proc/sys/net/ipv4/ip_forward
```

Linux Firewall

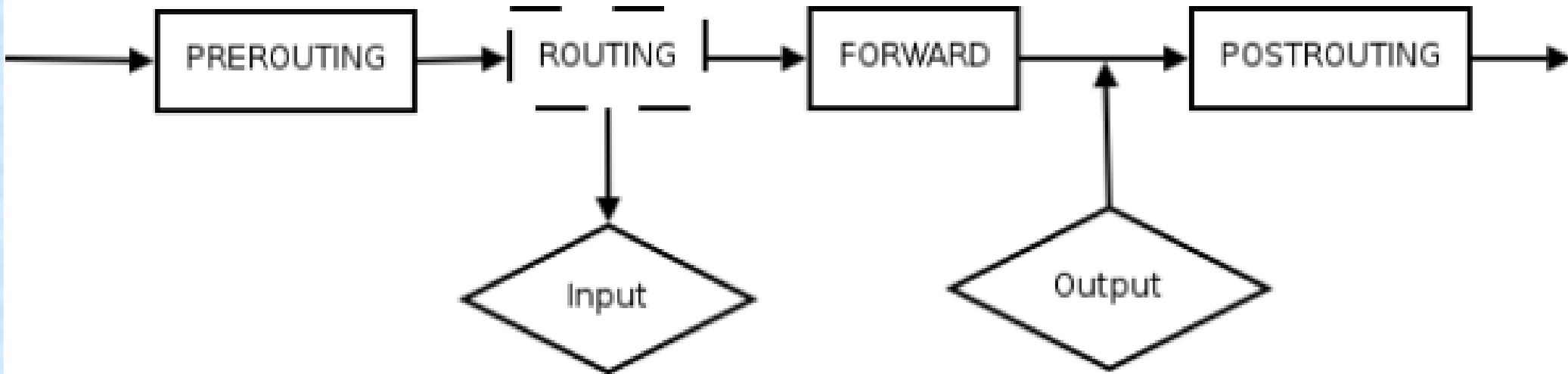
Il sistema di firewalling integrato nel kernel Linux si chiama netfilter, il tool in userspace per gestirlo si chiama iptables

netfilter ha 3 tabelle (=“ambiti”) di regole:

- * filter – serve per filtrare effettivamente i pacchetti
- * nat – serve per gestire le regole di NAT
- * mangle – serve per “modificare” i pacchetti

ogni tabella ha diverse chain (“catene”, ovvero collezioni di regole) che i pacchetti percorrono, e nelle quali viene scelta la loro sorte

Linux Firewall



PREROUTING:
direttive applicate
prima del processo
di routing.
(destination NAT)

FORWARD:
direttive applicate ai pacchetti
in transito

POSTROUTING:
direttive applicate dopo il
processo di routing.
(Source NAT / Masquerading)

INPUT e OUTPUT:
direttive applicate ai pacchetti destinati
o provenienti dall'host firewall

I pacchetti percorrono le varie regole presenti nelle catene, e:

- * Esiste una regola che corrisponde al pacchetto preso in esame: il destino del pacchetto è deciso dal target della regola
- * Non esiste una regola che corrisponde al pacchetto: il destino del pacchetto è deciso dalla politica di default della catena

attenzione: le regole sono a *corto-circuito*; al primo match viene deciso il destino del pacchetto.

iptables

La sintassi del comando iptables è molto varia, ci conviene vederla per esempi:

```
iptables -P OUTPUT ACCEPT
```

```
iptables -t nat -A POSTROUTING -o eth0 -j MASQUERADE
```

```
iptables -A INPUT -p tcp -s 192.168.1.1 --dport 22 -j ACCEPT
```

```
iptables -I 10 INPUT -p tcp -s ! 192.168.1.2 --dport 80 -j DROP
```

```
iptables -A OUTPUT -d 192.168.5.12 -j DROP
```

```
iptables -N lan1_to_lan2
```

```
iptables -P lan1_to_lan2 DROP
```

```
iptables -A lan1_to_lan2 -m state --state  
ESTABLISHED,RELATED -j ACCEPT
```

```
iptables -A lan1_to_lan2 -s 192.168.5.0/24 -d 192.168.6.0/24 -m  
state --state NEW -j ACCEPT
```

```
iptables -A FORWARD -i eth1 -o eth2 -j lan1_to_lan2
```

Ecco le principali opzioni (di gestione chain) del comando iptables:

- t <table> : usa la tabella <table>. se non usato si intende filter
- L : elenca tutte le regole
- A <chain> : append, aggiunge la regola in fondo alla catena
- I <chain> <pos> : inserisce la regola in posizione <pos>
- D <chain> <pos> : elimina la regola in posizione <pos>
- P <chain> <policy> : setta la politica di default per la catena. le politiche possono essere ACCEPT, DROP
- N <chain> : crea una nuova catena

In fondo ad ogni regola, il valore del parametro `-j` serve a definire il destino del pacchetto:

- j `<altra_chain>` : dice al pacchetto di percorrere `<altra_chain>`. Se in `<altra_chain>` non viene fatto nessun match, il pacchetto prosegue nella catena “chiamante”
- j `ACCEPT` : il pacchetto viene accettato
- j `DROP` : il pacchetto viene cestinato
- j `REJECT` : il pacchetto viene rimandato indietro. poco consigliato.
- j `RETURN` : il pacchetto viene mandato alla fine della catena
- j `LOG` : il pacchetto viene loggato. non a corto-circuito.

Nella tabella di nat si usa invece:

- j `MASQUERADE` : effettua il masquerading
- j `DNAT` : effettua il Destination NAT
- j `SNAT` : effettua il Source NAT

Ecco ora alcune opzioni per il matching dei pacchetti:

- o <iface> : l'interfaccia di rete in uscita è <iface>
- i <iface> : l'interfaccia di rete in ingresso è <iface>
- p <proto> : protocollo: tcp, udp, icmp, ...
- s <ip/net> : indirizzo/i ip sorgente
- d <ip/net> : indirizzo/i ip destinazione
- sport <port> : porta sorgente; solo tcp e udp
- dport <port> : porta destinazione; solo tcp e udp

iptables: principali opzioni

Oltre alle opzioni di base per il matching dei pacchetti, è possibile usare dei moduli che contengono opzioni più avanzate. L'uso di un modulo si definisce con `-m <module>`

ad esempio:

```
iptables ..... -m mac --mac-source ... -m state --  
state ... -j ...
```

iptables: principali moduli e opzioni

- m mac : effettua controlli sugli indirizzi MAC
 - mac-source <mac> : indirizzo mac sorgente
- m unclean : sanity check sui pacchetti
- m state : controlla lo stato dei pacchetti
 - state <STATES> : definisce lo stato dei pacchetti:
 - * NEW : il pacchetto crea una nuova connessione
 - * ESTABLISHED : appartiene a una conn. esistente
 - * RELATED : il pacchetto è correlato a una conn. esistente
 - * INVALID : non è possibile identificare il pacchetto
- m mark : controlla se il pacchetto è stato “marchiato”
 - mark <value>
- m tos : controlla il type of service
 - tos <tos> : legge il tos dall'header del pacchetto

continua...

iptables: principali moduli e opzioni

- m owner : (solo per OUTPUT), verifica l'user/group locale che ha originato i pacchetti
 - uid-owner <userid>
 - gid-owner <groupid>
 - pid-owner <processid>
- m ah
- m esp : lavorano sugli header pacchetti ipsec
- m ttl
 - ttl <value> : controlla il time-to-live
- m length
 - length <len> : verifica la lunghezza del pacchetto

Esempio di firewall per protezione locale

```
iptables -P OUTPUT ACCEPT
```

```
iptables -P INPUT DROP
```

```
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j  
ACCEPT
```

```
iptables -A INPUT -p tcp -s 192.168.1.91 --dport 22 -m state --state  
NEW -j ACCEPT
```

Esempio di firewall per semplice nat (e un minimo di protezione)

```
iptables -t nat -o $internet_iface -j MASQUERADE
```

```
iptables -P FORWARD DROP
```

```
iptables -N lan2internet
```

```
iptables -N internet2lan
```

```
iptables -A FORWARD -i $internet_iface -o $lan_iface -j  
internet2lan
```

```
iptables -A FORWARD -i $lan_iface -o $internet_iface -j  
lan2internet
```

```
iptables -A lan2internet -p tcp -d www.microsoft.com -j DROP
```

```
iptables -A lan2internet -j ACCEPT
```

```
iptables -A internet2lan -m state --state ESTABLISHED,RELATED  
-j ACCEPT
```

- date due reti (interna: 192.168.1.0/24 ed esterna: 192.168.200.0/24) progettare un firewall CHIUSO (policy di default: DROP) che:
- * mascheri gli ip interni con l'ip esterno
 - * blocchi tutto (tranne pacchetti relativi a connessioni già stabilite) da esterna a interna
 - * consenta l'ssh verso il firewall dall'ip 192.168.200.11
 - * consenta l'uscita verso l'esterno di connessioni http e https provenienti dalla rete interna

ecco come potrebbe essere...

```
if_esterna="eth0"
if_interna="eth1"
net_esterna="192.168.200.0/24"
net_interna="192.168.1.0/24"
iptables -N in2out
iptables -N out2in
iptables -t nat -A POSTROUTING -o $if_esterna -j MASQUERADE
iptables -P INPUT DROP
iptables -P FORWARD DROP
iptables -A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A INPUT -i $if_esterna -s 192.168.200.11 -p tcp --dport 22 -m state
    --state NEW -j ACCEPT
iptables -A FORWARD -i $if_esterna -s $net_interna -j DROP
iptables -A FORWARD -i $if_interna -s $net_esterna -j DROP
iptables -A FORWARD -i $if_esterna -o $if_interna -j out2in
iptables -A FORWARD -i $if_interna -o $if_esterna -j in2out
iptables -A out2in -m state --state ESTABLISHED,RELATED -j ACCEPT
iptables -A in2out -p tcp --dport 80 -m state --state NEW -j ACCEPT
iptables -A in2out -p tcp --dport 443 -m state --state NEW -j ACCEPT
```

Un proxy può essere utilizzato per

- * condividere una connessione internet tra più host
- * applicare policy con filtro accessi e contenuti
- * ottimizzare la banda disponibile

Essendo un proxy specifico per ogni protocollo a livello applicazione, esistono proxy http, proxy ftp, proxy pop3, proxy smtp, proxy ldap, ...

I protocolli cifrati TLS o SSL (https, ftps, pop3s, ...) non possono far uso di proxy, in quanto i pacchetti trasmessi sono cifrati e non possono essere quindi analizzati

Squid è un proxy http può operare in diversi modi:

- * proxy only
- * proxy + cache

per entrambi questi modi può lavorare in maniera trasparente (senza configurazione dei client),

MA, in questo caso:

- * niente https
- * niente autenticazione

proxy semitrasparente

Un'alternativa alla trasparenza è quella di usare i metodi di configurazione automatica del proxy; se questi sono abilitati, e il client si chiama *client1.miarete.miodominio.it*, il client cercherà un file di configurazione chiamato *wpad.dat* su <http://wpad.miarete.miodominio.it/wpad.dat>
<http://wpad.miodominio.it/wpad.dat>
etc...

vedere

http://faberlibertatis.org/wiki/Web_Proxy_Autodiscovery_Protocol

Squid e DansGuardian

Squid è principalmente un proxy http, ma può fungere, limitatamente, anche da proxy ftp.

DansGuardian è un potente sistema di content filtering che migliora la policy di accesso di un proxy verificando le richieste su una blacklist. Se la richiesta non viola i vincoli impostati nella configurazione, DansGuardian la dirotta al proxy permettendo il download della risorsa. La blacklist può essere scaricata da Internet e periodicamente aggiornata con una procedura automatica oppure manuale.

Squid e DansGuardian: installazione

```
apt-get install squid dansguardian
```

Configurazione di Squid

Squid è configurabile dal file */etc/squid/squid.conf*
Un facile metodo di configurazione è quello di togliere il simbolo di commento (#) alle direttive già presenti nel file, spiegate in maniera sufficientemente esaustiva.

Il file di configurazione è un elenco di direttive nella forma

direttiva [argomenti]

vediamo un esempio di configurazione:

squid.conf (1/5)

```
# Indirizzo IP e porta d'ascolto  
http_port 192.168.10.1:3128
```

```
# Pagine Web dinamiche da non memorizzare nella cache  
acl CGI urlpath_regex cgi-bin \?  
acl ASP urlpath_regex asp \?  
acl PHP urlpath_regex php \?  
acl JSP urlpath_regex jsp \?  
no_cache deny CGI ASP PHP JSP
```

```
# Dimensioni massime e minime delle  
# risorse da memorizzare in cache  
maximum_object_size 10240 KB  
minimum_object_size 10 KB
```

squid.conf (2/5)

```
# Identificazione delle reti locali a cui
# faranno riferimento le direttive d'accesso
acl all src 0.0.0.0/0.0.0.0
acl localnet src 192.168.10.0/255.255.255.0
acl localnet src 192.168.11.1-192.168.11.20/255.255.255.255
acl localhost src 127.0.0.1/255.255.255.255
acl manager proto cache_object

# Definizione dei metodi di connessione a cui
# faranno riferimento le direttive d'accesso
acl CONNECT method CONNECT
```

squid.conf (3/5)

```
# Identificazione delle porte TCP a cui
# faranno riferimento le direttive d'accesso
acl SSL_ports port 443 563
acl Safe_ports port 80          # http
acl Safe_ports port 21         # ftp
acl Safe_ports port 443 563    # https, snews
acl Safe_ports port 70        # gopher
acl Safe_ports port 210       # wais
acl Safe_ports port 1025-65535 # unregistered ports
acl Safe_ports port 280       # http-mgmt
acl Safe_ports port 488       # gss-http
acl Safe_ports port 591       # filemaker
acl Safe_ports port 777       # multilink http
acl Safe_ports port 901       # SWAT
```

```
# Definizione delle estensioni dei file a cui
# faranno riferimento specifiche direttive d'accesso
acl ban_mp3 url_regex -i \.mp3$
acl ban_exe url_regex -i \.exe$

# Identificazione di un file di testo contenente
# una lista di espressioni regolari a cui
# faranno riferimento le direttive d'accesso
acl forbidden url_regex "/usr/local/squid/etc/forbidden.txt"
```

squid.conf (5/5)

```
# Direttive d'accesso (http_access):  
# definiscono le regole di accesso al proxy  
http_access allow manager localhost  
http_access deny manager  
http_access deny !Safe_ports  
http_access deny CONNECT !SSL_ports  
http_access deny ban_mp3  
http_access deny ban_exe  
http_access deny forbidden  
http_access allow localnet  
http_access allow localhost      # permette l'accesso dal loopback
```

Le regole sono a corto-circuito,
l'ordine è quindi importante!

Due "acl" sulla stessa riga
sfruttano l'operazione AND

```
# Direttiva che vieta l'accesso a tutti i  
# casi non contemplati  
# Deve essere l'ultimo tag http_access  
http_access deny all
```

Come si può osservare, gran parte delle direttive prese in considerazione nell'esempio fanno capo a due forme:

- * **acl**
- * **http_access**

squid.conf: direttive *acl*

In generale identificano un contesto, come ad esempio una gamma di indirizzi di rete, un'espressione regolare contenuta negli URL, un tipo di porta, etc.. Ogni contesto definito è identificato con un nome che segue la parola chiave *acl*. La struttura sintattica di queste direttive pertanto è la seguente:

`acl nome tipo espressione`

Ad esempio, la direttiva

```
acl localnet src 192.168.10.0/255.255.255.0
```

identifica con il nome *localnet* un'intera sottorete. Per le postazioni che hanno un indirizzo di rete che rientra in questo contesto sarà specificata una direttiva d'accesso specifica.

squid.conf: tipi di *acl*

- * `acl nome src ip/mask` : discriminazione su ip sorgente
- * `acl nome dst ip/mask` : discriminazione su destinazione
- * `acl nome dstdomain .a.com` : discriminazione su dominio di destinazione
- * `acl nome dstdom_regex [-i] ...` : come sopra, ma regex
- * `acl nome time <days> <hours>` : a seconda di data/ora
- * `acl nome url_regex [-i] ^http://...` : a seconda di url
- * `acl nome urlpath_regex [-i] \.gif$` : a seconda di path nell'url
- * `acl nome port 80` : a seconda di porta destinazione
- * `acl nome method GET | POST` : a seconda di metodo
- * `acl nome browser [-i] regexp` : a seconda di User-Agent
- * `acl nome maxconn <num>` : a seconda del numero di connessioni
- * `acl fileupload req_mime_type -i ^multipart/form-data$`
- * `acl javascript rep_mime_type -i ^application/x-javascript$`

Impostano le regole d'accesso per ogni contesto definito in precedenza. In altri termini sono le direttive d'accesso che indicano al proxy come dovrà comportarsi caso per caso. La struttura sintattica di queste direttive è la seguente:

```
http_access azione nome_acl1 nome_acl2 ...
```

L'azione è definita da due parole chiave alternative: **deny** vieta l'accesso ad Internet nel contesto associato alla direttiva, **allow** lo permette. Nella configurazione di esempio sono specificate dieci direttive d'accesso.

Squid: avvio, arresto, ...

```
/etc/init.d/squid stop  
/etc/init.d/squid start
```

...

Squid e autenticazione LDAP

```
auth_param basic program /usr/lib/squid/ldap_auth -v 3 -b  
"dc=faberlibertatis,dc=org" -D cn=proxy,dc=faberlibertatis,dc=org  
-w password -f uid=%s 192.168.1.1  
auth_param basic children 5  
auth_param basic realm Web-Proxy  
auth_param basic credentialsttl 1 minute  
acl ldap-auth proxy_auth REQUIRED  
http_access deny !ldap-auth  
http_access allow localhost  
http_access deny all
```

Squid e autenticazione su file di testo

```
auth_param basic program /usr/lib/squid/nrsa_auth
/etc/squid/passwd
auth_param basic children 5
auth_param basic realm Squid proxy-caching web server
auth_param basic credentialsttl 2 hours
auth_param basic casesensitive off

acl proxy-auth proxy_auth REQUIRED
http_access deny !proxy-auth
http_access allow localhost
http_access deny all
```

Ci sono due possibili modi per integrare DansGuardian con Squid:

- * far arrivare le richieste dei client direttamente a DansGuardian, in caso di accesso consentito DansGuardian rimanderà la connessione a Squid (contro: autenticazione utenti limitata, nessun possibile filtraggio su ip sorgenti (sorgente sempre **127.0.0.1!**))
- * usare DansGuardian come Parent-Cache di Squid

ovviamente vedremo la soluzione “Parent-Cache”!

Squid, DansGuardian, Squid

* Inserire in Squid un'altra porta su cui rimanere in ascolto, avremo quindi

```
http_port 192.168.1.1:3128
```

```
http_port 127.0.0.1:8081
```

* Acl

```
acl my_ips dst 192.168.1.1
```

* Inserire direttive Cache-Peer

```
cache_peer 127.0.0.1 parent 8080 0 no-query no-digest no-  
netdb-exchange
```

```
cache_peer_access 127.0.0.1 deny localhost
```

```
cache_peer_access 127.0.0.1 allow all
```

```
always_direct allow my_ips
```

```
never_direct allow all !localhost
```

* La prima direttiva http_access dovrà essere

```
http_access allow localhost
```

Configurazione base di DansGuardian

/etc/dansguardian/dansguardian.conf

Network Settings

filterip = 127.0.0.1

filterport = 8080

proxyip = 127.0.0.1

proxyport = 8081

`/etc/dansguardian/dansguardian.conf`

La sintassi non si discosta da quella di squid.conf ad eccezione dell'inserimento del segno uguale fra il nome della direttiva e il valore attribuito:

`direttiva = [valore]`

nel file `/etc/dansguardian/dansguardianf1.conf`

sono invece definiti tutti i vari filtri

dansguardian.conf (1/3)

```
# Azione eseguita in caso di accesso negato  
reportinglevel = 3
```

```
# Directory dei file HTML  
# restituiti in caso di accesso negato  
languagedir = '/etc/dansguardian/languages'
```

```
# Lingua utilizzata dalla directory  
# languagedir (default: ukenglish)  
language = 'italian'
```

```
# Formato del file di registro  
# L'opzione 3 imposta lo stesso formato  
# del file access.log di Squid  
logfileformat = 1
```

```
# Percorso del file di registro  
loglocation = '/var/log/dansguardian/access.log'
```

dansguardian.conf (2/3)

Indirizzo di rete di DansGuardian
filterip = 192.168.10.1

Porta di ascolto
filterport = 8080

Indirizzo di rete del server proxy
E' predefinito l'indirizzo di loopback
ipotizzando che proxy e filtro dei contenuti
siano installati sullo stesso host
proxyip = 127.0.0.1

Porta di ascolto del proxy
proxyport = 8081

URL di reindirizzamento se l'accesso negato reindirizza ad un CGI
Lasciare l'impostazione predefinita o commentare se si usa il template
accessdeniedaddress = 'http://myserver/cgi-bin/dansguardian.pl'

dansguardian.conf (3/3)

```
# Metodo di ponderazione delle pagine  
# il valore predefinito 2 fa sì che una  
# frase sia conteggiata una sola volta  
# se riscontrata nella pagina  
weightedphrasemode = 2
```

```
# Creazione di una cache dei file e  
# degli indirizzi negati. L'impostazione  
# predefinita (on) permette di triplicare  
# la velocità del processo di scansione,  
# pertanto è raccomandata se il sistema  
# è in esecuzione su un  
# computer lento  
createlistcachefiles = on
```

Per evitare tempi di avvio
colossali, modifichiamo
qui e in
`/etc/dansguardianf1.conf`
`virusscan = on`
in
`virusscan = off`

DansGuardian Blacklist

Le blacklist sono il fulcro del servizio di filtering, esse includono indirizzi ip, url, espressioni regolari, ... e sono aggiornate molto frequentemente, in modo da seguire il “dinamismo” di internet.

La blacklist

<http://urlblacklist.com/?sec=download>

(14 mega di file di testo compressi, non è fuffa :-)

contiene anche numerosi siti italiani, quindi è ottima per i nostri scopi.

Per installarla, scompattiamo il tar in */etc/dansguardian*

```
# tar xzvf bigblacklist.tar.gz
```

DansGuardian e Blacklist

L'operazione di scompattamento crea una directory contenente gli archivi di url e domini (file di testo) distribuiti in varie directory a seconda della categoria.

Per usufruire delle blacklist è necessario configurare DansGuardian specificando per quali categorie di blacklist deve essere attivato il filtro.

L'abilitazione delle categorie si effettua configurando alcuni file contenuti nella directory */etc/dansguardian*, in particolare i file *bannedsitelist* (relativo ai domini) e *bannedulrllist* (relativo a specifici file o percorsi). Questi file sono predisposti per l'inclusione di tutte le categorie. Per attivare il filtro su una specifica categoria è sufficiente rimuovere il carattere di commento #.

DansGuardian e Blacklist

Ad esempio, togliendo il commento alla seguente direttiva (riportata in *bannedsitelist*), DansGuardian bloccherà tutti i domini elencati nel file domains contenuto in */etc/dansguardian/blacklists/warez*:

```
.Include</etc/dansguardian/blacklists/warez/domains>
```

avvio e arresto di DansGuardian

```
/etc/init.d/dansguardian start  
/etc/init.d/dansguardian stop  
/etc/init.d/dansguardian reload
```

...

DansGuardian in action...

