Incontri a Tema

Universita' degli Studi di Padova

GNU/Linux Firewall

Alberto Cammozzo – Mauro Malvestio
{mmzz, malveo} @ stat.unipd.it

NAT (SNAT / DNAT / MASQ)

- SNAT : Cambia la provenienza della connesione [SOURCE ADDR], viene eseguito nella fase di post-routing :

  Usato spesso in contesti "routed" per forzare il traffico verso un delineato gw :
  Es. Due reti (vlan) distinte, una per la rete wifi e una per le macchine non manager, con addr rispettivi (LAN1)192.168.1.x, (LAN2)192.168.2.x, con gw diversi.

```
$ $IPTABLES -A POSTROUTING -t nat -s $LAN1 -m state --state new  -j SNAT --to-source 147.162.35.10
$ $IPTABLES -A POSTROUTING -t nat -s $LAN2 -m state --state new  -j SNAT --to-source 147.162.35.11
```

- DNAT, Cambia la destinazione della connessione [DESTINATION ADDR], viene eseguito nella fase di pre-routing :

  - ✓ Proxy Trasparente (Squid)
  - ✓ Port Forwarding (DMZ)
  - ✓ Load Balancing/Sharing (Linux VirtualServer)

- MASQUERADE, e' una forma elaborata di SNAT, in quanto non lavora su [SOURCE ADDR] ma direttamente sull'interfaccia, e quello che tutti chiamano semplicemente NAT.

```
$ $IPTABLES -t nat -A POSTROUTING –o eth1 -j MASQUERADE
```

Connection Tracking

Il connection tracking mantiene una tabella nel proc filesystem (/proc/net/ip_contrack) in cui tiene traccia dello stato della connessione tramite il modulo del kernel “ip_contrack”

```
tcp  6 431999 ESTABLISHED src=147.162.35.15 dst=147.162.35.254 sport=49156 dport=22 packets=1395 bytes=94284
     src=147.162.35.254 dst=147.162.35.15 sport=22 dport=49156 packets=987 bytes=363019 [ASSURED] mark=0 use=1
```

Il conntrack va usato in conbinazione del modulo “xt_state” , utilizzando *iptables* con lo switch `–state` :

- ✓ ***NEW*** : PKG (SYN TCP) o UDP
- ✓ **ESTABLISHED** : Pacchetti relativi a connessioni già stabilite
- ✓ **RELATED** : Pacchetti correlati a connessioni esistenti ed established. Es. FTP (controllo e dati)
- ✓ **INVALID** : Pacchetti che non rientrano in alcuno dei suddetti stati, di solito vengono droppati

Protezione dell’interfaccia di management ssh di un host linux:

```
$IPTABLES -P INPUT DROP
$IPTABLES -A $CHAIN -i $MANAGEMENT_INTERFACE -m state --state INVALID -j LOG --log-prefix "DROP INVALID" --log-
ip-options --log-tcp-options
$IPTABLES -A $CHAIN -i $MANAGEMENT_INTERFACE -m state --state INVALID -j DROP
$IPTABLES -A $CHAIN -m state --state ESTABLISHED,RELATED -j ACCEPT
$IPTABLES -A $CHAIN  -p tcp –s $ADMIN_HOST --dport ssh --syn -m state --state NEW -j ACCEPT
$IPTABLES -A $CHAIN -i ! lo -j LOG --log-prefix "DROP " --log-ip-options --log-tcp-options

$IPTABBLES  $MAX_CONN  > /proc/sys/net/ipv4/netfilter/ip_conntrack_max
```

Proxy ARP

E’ una tecnica di utilizzo del protocollo ARP per fornire un meccanismo *ad hoc* di routing, che non richiede la configurazione dell'indirizzo IP del router sugli host, si ottiene lo stesso risultato della modalita’ bridge :

Es. Cisco GW sul 192.168.0.253:00:13:10:92:9a:57 [eth0 Auto-MDX] , LAN su 192.168.0.xxx [eth1]

```
for interface in lo eth0 eth1
do
      ip route flush dev $interface
      ip addr flush dev $interface
      ip link set down dev $interface
done

ip address add 127.0.0.0/8 dev lo
ip address add 192.168.0.1/32 dev eth0
ip address add 192.168.0.1/32 dev eth1

for interface in lo eth0 eth1
do
      ip link set up dev $interface
done

echo 1 > /proc/sys/net/ipv4/conf/eth0/proxy_arp
echo 1 > /proc/sys/net/ipv4/conf/eth1/proxy_arp

ip route add 127.0.0.0/8 dev lo
ip route add 192.168.0.254/32 dev eth0
ip route add 192.168.0.0/24 dev eth1
ip route add default via 192.168.0.254

ip neigh add proxy 192.168.0.254 lladdr 00:13:10:92:9a:57  nud permanent eth1

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

Squid Proxy Trasparente [1]

Il proxy server Squid (http://www.squid-cache.org), permette nella modalita' trasparente il filtering a livello http solo per il protocollo www (80), non permette di controllare il traffico https (443) e l'autenticazione.

```
$ cat /etc/squid/squid-trasparent.conf

http_port 147.162.35.xxx:3128 transparent
cache_dir null /tmp
cache_access_log none
cache_store_log none
cache_log /dev/null
cache_mem 64 MB
acl statistica src 147.162.23.0/24 147.162.223.192/26 147.162.214.128/27 147.162.35.0/24
acl studio src 147.162.35.51-147.162.35.62
acl permitted dstdomain www.unipd.it www.statistica.unipd.it
acl aule src 147.162.35.15 147.162.35.45 147.162.35.101-147.162.35.167 147.162.35.181-147.162.35.198
                        147.162.35.30-147.162.35.39  147.162.35.51-147.162.35.62
acl malware_block_list url_regex -i "/etc/squid/malware_block_list.txt"
acl im url_regex -i "/etc/squid/im.acl“
http_access deny statistica malware_block_list
deny_info ERR_DENY_MALWARE malware_block_list
http_access deny im aule
deny_info ERR_DENY_SERVICE aule

http_access allow permitted
http_access deny studio
deny_info ERR_DENY_SERVICE studio

http_access allow statistica
```

Squid Proxy Trasparente [2]

```
$ cat /etc/squid/malware_block_list.txt

^http\:\/\/(.+@)?mensagemtim\.rbcmail\.ru\/
^http\:\/\/(.+@)?mensagenscartaonatal\.zxq\.net\/natal\/
^http\:\/\/(.+@)?xscanner\.malwarealarm\.com\/a\/
^http\:\/\/(.+@)?xscanner\.spyshredder-scanner\.com\/a\

$ cat /etc/squid/im_block_list.acl
.icq.com
^application/x-msn-messenger$
.msg.yahoo.com webmessenger
.webmessenger .messenger.*
messenger.yahoo
gateway.dll
.google.com/talk*

$ CHAIN=PREROUTING

$ $IPTABLES -t nat -A $CHAIN -i int.5 -m iprange --src-range 147.162.35.101-147.162.35.167 \
       -m tcp -p tcp --dport 80 -m state --state new -j REDIRECT --to-ports 3128

$ $IPTABLES -t nat -A $CHAIN -i int.2 -s 147.162.23.0/24  -m tcp -p tcp --dport 80 \
       -m state --state new -j REDIRECT --to-ports 3128
```

Squid Autenticazione [1]

```
http_port 192.168.2.254:3128
acl ateneo_networks dst 147.162.0.0/255.255.0.0
acl ateneo_websites dstdomain www.unipd.it

auth_param basic program /usr/lib/squid/yak-multi.pl
auth_param basic children 5
auth_param basic realm Authentication U:Matricola P:PIN, U:nome.cognome@unipd.it P:PASSWORD
auth_param basic credentialsttl 1 hours

acl AuthorizedUsers proxy_auth REQUIRED
http_access allow ateneo_networks
http_access allow ateneo_websites
http_access allow all AuthorizedUsers

http_access deny all
http_reply_access allow all
cache_effective_group proxy
via off
forwarded_for off
```

Squid Autenticazione [2]

I moduli di autenticazioni di Squid sono diversi {ldap, smb, pam}, per scrivere il vs dovete tenere presente :

1. Il login errato deve restituire “ERR”
2. Il login corrette deve restituire “OK”
3. L’autenticazione e’ un : while (1) { … , return “OK” OR return “ERR” }

```
while (<>) {
    chop;
    my ($u,$p) = split;
    my $ans = 'ERR';
    if( $u =~ /^[0-9]/ && $p =~ /^[0-9]/ ) {
        $ans = 'OK' if ($u && $p && check_http(substr($u,0,6),substr($p,0,5),$sis_url));
    }
    if( $u =~ /^[a-z]/i ) {
        if ( $enable_pop ) {
            $ans = 'OK' if ($u && $p && check_pop(substr($u,0,30),substr($p,0,7)));
        }
        else {
            $ans = 'OK' if ($u && $p && check_http(substr($u,0,30),substr($p,0,7),$webmail_url));
        }
    }
    print "$ans\n";
}
```

Modulo Perl completo disponibile su : http://foss.stat.unipd.it/mediawiki/index.php/Squid_YAK-Auth_POPS/HTTPS

Squid Autenticazione [3]

```
CHAIN=JAIL
$IPTABLES -N $CHAIN
$IPTABLES -A FORWARD -j $CHAIN
$IPTABLES -F $CHAIN
$IPTABLES -A $CHAIN -i int.10 -s $LAN10 -d $SQUID_JAIL -m tcp -p tcp -m multiport --dports $SQUID_PORT \
                    -m state --state NEW -j ACCEPT
$IPTABLES -A $CHAIN -i int.10 -s $LAN10 -m tcp -p tcp -m multiport --dports ssh,imap,pop3,imaps,pop3s \
                    -m state --state NEW -j ACCEPT

for DNS in $DNS_SERVERS ; do
        $IPTABLES -A $CHAIN -i int.10 -s $LAN10 -d $DNS -m udp -p udp -m multiport --dports domain -m state --state NEW -j ACCEPT
Done

for YAK in $YAK_OPAC ; do
        $IPTABLES -A $CHAIN -i int.10 -s $YAK -d ! $CATALOGO -j REJECT
Done
```

```
root@fw:~# iptables -nL JAIL
Chain JAIL (1 references)
target     prot opt source               destination
ACCEPT     tcp  --  192.168.2.0/24       192.168.2.254        tcp multiport dports 3128 state NEW
ACCEPT     tcp  --  192.168.2.0/24       0.0.0.0/0            tcp multiport dports 22,143,110,993,995 state NEW
ACCEPT     udp  --  192.168.2.0/24       147.162.35.1         udp multiport dports 53 state NEW
ACCEPT     udp  --  192.168.2.0/24       147.162.1.2          udp multiport dports 53 state NEW
REJECT     all  --  192.168.2.10        !147.162.210.147      reject-with icmp-port-unreachable
REJECT     all  --  192.168.2.99        !147.162.210.147      reject-with icmp-port-unreachable
```

VLAN

Linux supporta in kernel lo standard 802.11q, supporta le VLAN (Tagged e Un-Tagged).

Lato Switch (ex. IOS Cisco) :

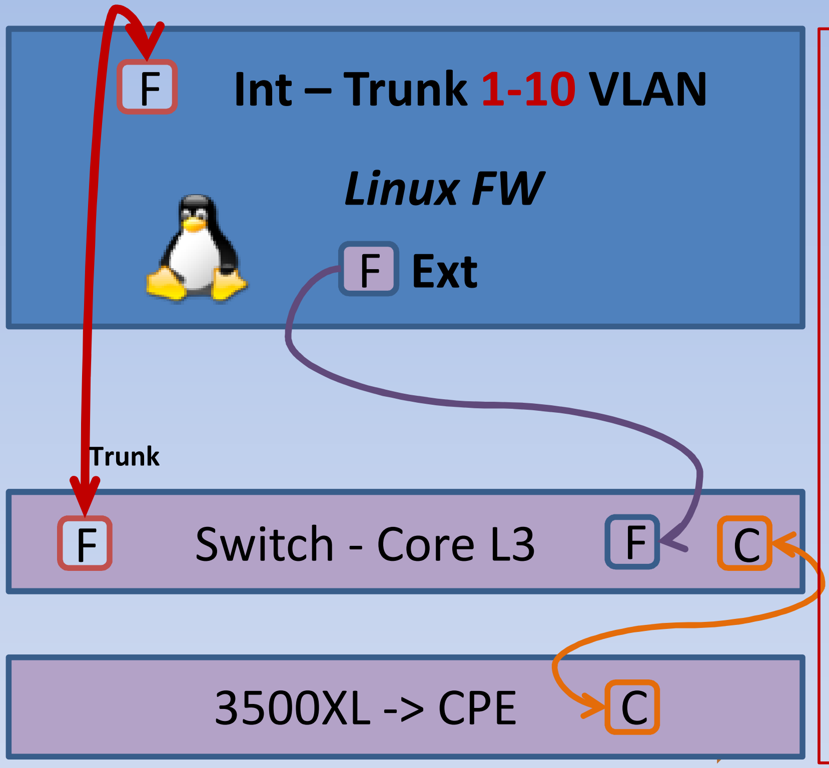
```
sh running-config interface fastEthernet 0/36
Building configuration...
Current configuration:
!
interface FastEthernet0/36
         duplex full
         speed 100
         switchport trunk encapsulation dot1q
         switchport trunk allowed vlan 1-5,1002-1005
         switchport mode trunk
         spanning-tree portfast
End
```

```
# Aggiunge all’interfaccia $IFACE la VLAN con tag $TAG
$ vconfig add $IFACE $TAG
$ vconfig add int 2

$ ip a l int.2
2: int.2@int: <BROADCAST,MULTICAST,UP,10000> mtu 1500 qdisc noqueue
         link/ether 00:e0:81:04:44:88 brd ff:ff:ff:ff:ff:ff
         inet 147.162.23.254/24 brd 147.162.23.255 scope global int.2
         inet 147.162.23.225/24 brd 147.162.23.255 scope global secondary int.

$ cat /proc/net/vlan/config
VLAN Dev name    | VLAN ID
Name-Type: VLAN_NAME_TYPE_RAW_PLUS_VID_NO_PAD
int.2          | 2  | int
```

VLAN + Routing



```
$ cat /proc/net/vlan/config
VLAN Dev name     | VLAN ID
Name-Type: VLAN_NAME_TYPE_RAW_PLUS_VID_NO_PAD
int.2          | 2  | int
int.3          | 3  | int
int.4          | 4  | int
int.5          | 5  | int
int.6          | 6  | int
int.7          | 7  | int
int.10         | 10  | int

$ ip route list
147.162.252.112/29 dev ext  proto kernel  scope link  src 147.162.252.118
192.168.1.168/29 dev int.6  proto kernel  scope link  src 192.168.1.174
147.162.214.128/27 dev int.4  proto kernel  scope link  src 147.162.214.158
147.162.223.192/26 dev int.3  proto kernel  scope link  src 147.162.223.254
147.162.23.0/24 dev int.2  proto kernel  scope link  src 147.162.23.254
147.162.35.0/24 dev int.5  proto kernel  scope link  src 147.162.35.254
192.168.2.0/24 dev int.10  proto kernel  scope link  src 192.168.2.254
172.16.0.0/24 dev int.7  proto kernel  scope link  src 172.16.0.254
10.0.0.0/8 dev eth0  proto kernel  scope link  src 10.0.0.1
default via 147.162.252.117 dev ext
```

NetFilter Sicurezza

- Syn-Flood

```
$IPTABLES -A INPUT -m state --state NEW -p tcp -m tcp –syn -m recent --name synflood –set
$IPTABLES -A INPUT -m state --state NEW -p tcp -m tcp –syn  -m recent --name synflood –update \
--seconds 1 --hitcount 60 -j DROP
```

- Spoofed Packet

```
$ SPOOFED=”0.0.0.0/8 127.0.0.0/8 10.0.0.0/8 172.16.0.0/12 192.168.0.0/16 224.0.0.0/3 239.255.255.0/24
255.255.255.255”
$ for ip in $SPOOFED do
       $IPTABLES -A INPUT -s $ip –REJECT
       $IPTABLES -A INPUT -d $ip –REJECT
done
$ sysctl -w net.ipv4.conf.all.rp_filter=1
```

- Ping… yes ok, but not ping –f… (Smurfing…)

```
$IPTABLES -A INPUT -p icmp -m icmp --icmp-type address-mask-request -j DROP
$IPTABLES -A INPUT -p icmp -m icmp --icmp-type timestamp-request -j DROP
$IPTABLES -A INPUT -p icmp -m icmp -m limit --limit 1/second -j ACCEPT
```

- Bogus Packet

```
$IPTABLES -A INPUT -m state --state INVALID -j DROP
$IPTABLES -A FORWARD -m state --state INVALID -j DROP
$IPTABLES -A OUTPUT -m state --state INVALID -j DROP
$IPTABLES -A INPUT -p tcp -m tcp --tcp-flags SYN,FIN SYN,FIN -j DROP
$IPTABLES -A INPUT -p tcp -m tcp --tcp-flags SYN,RST SYN,RST -j DROP
```

NetFilter Moduli

- **Connlimit**
  - ✓ *$IPTABLES -A INPUT -p tcp --dport ssh --syn -m connlimit --connlimit-above 2 -j DROP*
- **Quota**
  - ✓ *$IPTABLES -A INPUT -m quota –quota $BYTES -d $D_ADDR -j ACCEPT*
- **Time**
  - ✓ *$IPTABLES -A INPUT –I $WIFI –s $WIFI_LAN -m time --timestart 22:00 –timestop 08:30 --days Mon,Tue,Wed,Thu,Fri -j ACCEPT*
- **Psd**
  - ✓ *$IPTABLES -A INPUT -m psd -j LOG --log-prefix “PORTSCAN!!!”*
- **Condition**
  - ✓ *$IPTABLES -A INPUT -m condition --condition ssh -p tcp –dport 22 -j ACCEPT*

  ```
  $ echo 0 || 1 > /proc/net/ipt_condition/ssh
  ```
- **String**
  - ✓ *$IPTABLES -A INPUT -p tcp -d $SQL_SERVER-m string --string “SQLDMO.SQLServer” -j LOG*
  - ✓ *$IPTABLES -A INPUT -p tcp -d $SQL_SERVER-m string --string “SQLDMO.SQLServer” -j REJECT --reject-with icmp-port-unreachable*

    Microsoft SQL Server Distributed Management Objects Buffer Overflow (*http://www.milw0rm.com/exploits/4398*)

Debian / RH / Slackware

Debian :

```
$ apt-get install iptables
```

/sbin/iptables – The administration utility/binary.

Redhat :

```
$ up2date -u iptables
$ yum install iptables
```

/etc/init.d/iptables – INIT script to start|stop|restart the service (and save rulesets).
/etc/sysconfig/iptables – RedHat's file for the iptables-save counter files (i.e. The saved rulesets).
/sbin/iptables – The administration utility/binary.

Slackware :

```
$ swaret –update && swaret –install iptables
```

/etc/rc.d/rc.firewall
/sbin/iptables – The administration utility/binary

Links

- http://www.sjdjweis.com/linux/proxyarp/
- http://www.acmeconsulting.it/Squid-Book/HTML/
- http://foss.stat.unipd.it/mediawiki/index.php/802.3Q_Vlan
- http://www.linuxdevcenter.com/pub/a/linux/2001/08/09/authen_squid.html