INTRODUZIONE

Quando un PC è connesso ad internet esso diventa, a tutti gli effetti, un nodo della rete. Il sistema connesso, può esporre dei servizi di rete, cioè delle applicazioni che hanno delle funzionalità specifiche e che rimangono in ascolto su una determinata porta (ad es. un server ftp o telnet o anche il classico servizio di condivisione dei file e delle stampanti di Windows).

In alcuni casi può capitare che questi servizi nascondano al loro interno delle vulnerabilità o, comunque, rendano manifesti dei difetti di configurazione tali da poter essere sfruttati per guadagnare l'accesso non autorizzato ad un sistema.

In genere prima di tentare una qualsiasi forma di intrusione un aggressore fa uso di software specifico per effettuare la scansione del target alla ricerca di eventuali porte in ascolto. Una volta individuate queste porte è molto facile risalire al tipo di applicazione in esecuzione e, da qui, al genere di problemi di sicurezza che affliggono l'applicazione stessa. Di conseguenza, diventa possibile sfruttare un exploit (una tecnica di attacco particolare che si basa sulla presenza di vulnerabilità note) in modo da ottenere l'accesso al sistema.

L'uso di un firewall, in combinazione con altri strumenti come l'IDS, può effettivamente garantire un livello di protezione discreto non soltanto contro i tentativi di sfruttare vulnerabilità più o meno note di un servizio, ma anche e soprattutto contro l'attività di scansione delle porte che normalmente costituisce sempre il preludio di un attacco.

Lo scopo di questo progetto e quello di realizzare un sistema firewall/IDS in grado di garantire un'adeguata protezione, alle due reti LAN presenti dietro il firewall.

La tesi realizzata in base a tale progetto, è divisa in quattro parti che riguardano, la corretta configurazione ed implementazione di diversi firewall open-source, l'installazione e la configurazione di un'IDS postro dietro al firewall in grado di rilevare eventuali attacchi che sono riusciti a oltrepassarlo, il traffic-shaping utilizzato per limitare la banda assegnata, in modo tale che il taffico, in ingresso e in uscita dalle LAN, non vada a saturare la banda disponibile; e infine dei test realizzati per provare l'effettiva sicurezza del sistema creato.

CAPITOLO 1

PANORAMICA SUI PROBLEMI DI SICUREZZA

1.0 Introduzione ai firewall

Il mondo attuale delle reti e dei computer può presentare diverse minacce per quanto riguarda la sicurezza. Un utente irresponsabile è in grado di provocare ad un sistema non adeguatamente protetto danni enormi e l'interruzione dei servizi offerti.

La sicurezza riguardante le reti informatiche sta necessariamente assumendo un ruolo ed un'importanza predominante dovuta all'aumento vertiginoso degli utenti che hanno a disposizione un collegamento ad internet.

Per proteggere una qualsiasi rete accessibile dall'esterno, un amministratore di rete dovrà necessariamente pianificare adeguatamente la struttura da utilizzare e adottare tutte le misure di sicurezza atte a garantire l'integrità dei dati presenti sulle proprie macchine e l'accessibilità ai servizi offerti.

Molti di questi obiettivi possono essere raggiunti utilizzando un sistema di firewalling che realizza due funzioni: una implementa una politica ben precisa nei confronti delle diverse connessioni tra la propria rete e quella esterna, l'altra memorizza tutte le informazioni di interesse attraverso i file di log.

Uno dei sistemi software più utilizzati per la realizzazione di un firewall, è il programma IPTables, diffuso in ambiente GNU/Linux, che possiede, tra le tante altre caratteristiche, anche un sistema di filtraggio dei pacchetti (packet filtring) molto efficace.

1.1 Il concetto di sicurezza

All'interno di un mondo quale quello della comunicazione digitale in continua evoluzione e che presenta costi sempre più bassi di interconnessione, il concetto di sicurezza sta assumendo un ruolo sempre più importante, ed è diventato un requisito fondamentale per ogni software, sistema, rete semplice o complessa. La sicurezza di rete sta assumendo giorno dopo giorno un ruolo predominante all'interno di tali problematiche. Difatti, ogni comunicazione che attraversa internet o in generale una qualsiasi rete geografica, può toccare diversi nodi (host), dando quindi ad altri utenti l'opportunità di tracciare, intercettare, modificare i dati in transito. Un'altra problematica fondamentale consiste nell'evitare accessi non autorizzati alla propria rete da parte di malintenzionati in grado di appropriarsi, modificare o eliminare dati sensibili.

Un ipotetico attaccante potrebbe compromettere la sicurezza di una rete utilizzando diverse metodologie oramai note. Innanzitutto potrebbe, ad esempio, prendere il controllo di un host sfruttando un bug di qualche servizio che gira su tale macchina; successivamente, potrebbe esplorare il traffico della rete attraverso la tecnica del packet sniffing (cercando cioè di intercettare il più alto numero possibile di pacchetti che transitano attraverso il mezzo fisico) e successivamente utilizzare le macchine compromesse per attacchi di tipo DoS o DDoS in modo tale da occupare tutte le risorse messe a disposizione dai servizi della macchina vittima affinchè questa non sia più in grado di rispondere alle richieste legittime.

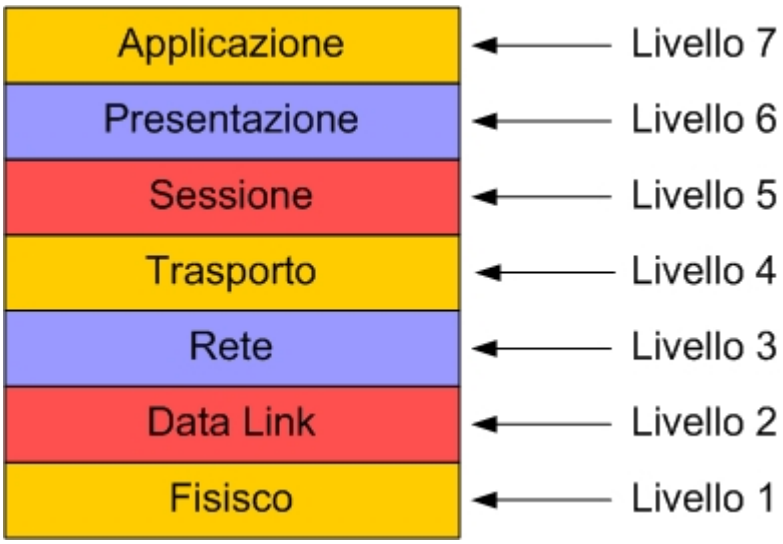
Per difendere la propria rete da tali attacchi e da molti altri è necessaria la conoscenza del protocollo che viene comunemente utilizzato per veicolare i pacchetti su internet, il TCP/IP.

1.2 Il protocollo TCP/IP

Una rete può essere idealmente definita come una maglia di collegamenti. Ogni nodo di questa struttura è generalmente un elaboratore. I diversi host sono in grado di comunicare attraverso dei pacchetti che rappresentano il mezzo con cui i dati possono transitare attraverso la rete. In ogni pacchetto è specificato il mittente e il destinatario dello stesso e la sua struttura dipende dalla rete fisica utilizzata.

I pacchetti vengono inviati e ricevuti in base a delle regole definite da un protocollo di comunicazione. Principalmente esistono due tipologie di protocolli a secondo se esso sia in grado di assicurare una connessione virtuale, mettendo a disposizione ad esempio, strumenti quali conferma dell'invio di un messaggio, ritrasmissione in caso di errore, controllo di flusso e ricomposizione dell'ordine dei pacchetti o meno.

Il nome TCP/IP rappresenta un insieme di protocolli di comunicazione basati su IP (*Internet Protocol*), un protocollo che si colloca all'interno del terzo livello ISO/OSI, il Network Level.



Il TCP (*Trasmission Control Protocol*), l'UDP (*User Datagram Protocol*) e l'ICMP (*Internet Control Message Protocol*) sono tutti protocolli di livello superiore che utilizzano l'IP incapsulando i propri messaggi all'interno dei pacchetti di tale protocollo. A loro volta, altri protocolli quali l'SMTP, l'FTP, l'HTTP utilizzeranno il TCP, l'UDP, l'ICMP, ponendosi quindi ad un livello superiore rispetto a questi, per la precisione a livello applicazione (livello 7 ISO/OSI corrispondente al livello 4 della pila internet).

Il datagramma IP è costituito da un'intestazione (header del pacchetto) e dai dati veri e propri nei quali possono anche essere incapsulate le informazioni di protocolli di livello superiore.

Tipicamente nell'header vengono specificate diverse informazioni che verranno utilizzate da tutti i nodi della rete coinvolti nel transito del pacchetto, per effetturare un corretto controllo di flusso, indirizzamento e controllo di integrità dei dati, quali la lunghezza del datagramma, la versione del protocollo utilizzato, il TTL (*Time to Live*), alcuni bit di checksum, gli indirizzi di sorgente e destinazione. Tali indirizzi sono composti da una sequenza di 32 bit suddivisi convenzionalmente in quattro gruppi da 8.

Mentre protocolli quali l'UDP e l'ICMP si basano su un servizio di distribuzione dei pacchetti privo di connessione e non affidabile, il TCP fornisce una serie di meccanismi in grado di garantire queste caratteristiche creando una sorta di connessione virtuale point-to-point tra i due elaboratori coinvolti nella comunicazione. Tutto ciò viene realizzato secondo un meccanismo chiamato *positive acknowledgemente with retrasmission* e da un *three-way handshake* iniziale. Inoltre, sia in TCP che in UDP, è rilevante il concetto di port, utilizzata per distinguere le diverse connessioni tra due host. La comunicazione tra l'implementazione del protocollo sul sistema operativo e l'applicativo che si serve di tali procedure, avverrà proprio attraverso questo meccanismo.

1.3 Il firewall

Un firewall è un sistema o un gruppo di sistemi che realizza una metodologia di controllo degli accessi tra due o più reti.

Può essere schematizzato attraverso una coppia di meccanismi: uno che realizza il blocco del traffico, l'altro che lo accetta. Alcuni firewall ripongono maggiore enfasi sul primo aspetto, altri sul secondo.

La prerogativa fondamentale consiste nel decidere la politica di sicurezza da adottare prima della realizzazione fisica della rete. È infatti di fondamentale importanza che l'amministratore sappia con assoluta precisione quali e quanti servizi offrire al mondo esterno attraverso la propria rete, quali privilegi concedere agli utilizzatori dei terminali interni e come strutturare fisicamente la rete stessa.

Spesso un sistema di protezione quale un firewall risulta complesso da configurare e richiede delle conoscenze avanzate che riguardano la struttura intrinseca e il funzionamento della rete, la conoscenza dei protocolli, i meccanismi di comunicazione tra le diverse tipologie di reti.

1.3.1 I vantaggi

Secondo un principio generale, un firewall viene realizzato sia per concedere alle postazioni interne un numero limitato di operazioni (sarà l'amministratore a dover valutare la fiducia che ripone nei proprio utenti), sia, soprattutto, per proteggere la rete interna da connessioni non autorizzate provenienti dal mondo esterno.

Un altro elemento di forza viene realizzato progettando una struttura di rete che porti il firewall ad essere l'unico punto di contatto tra la rete locale privata e la rete esterna. In tal modo sarà possibile mettere a disposizione un importante

meccanismo di logging degli accessi e del transito delle informazioni oltre a permettere un effettivo controllo del traffico nelle due direzioni.

Un firewall mette inoltre a disposizione dell'amministratore di rete dei dati molto importanti quali il traffico generato dalla rete stessa e insieme all'IDS, l'elenco dei tentativi di intrusione.

1.3.2 I punti deboli

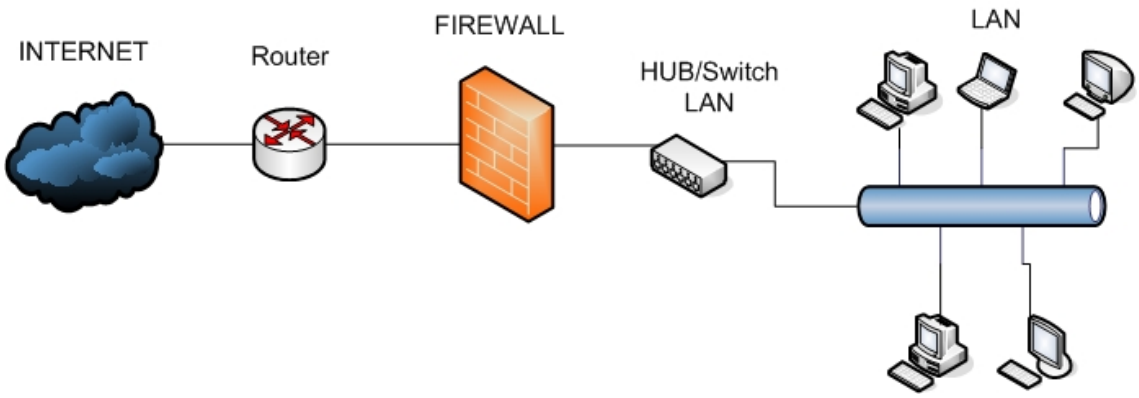
Un sistema illustrato in questi termini, presenta però alcuni punti deboli da non sottovalutare: infatti, un firewall non è ovviamente in grado di proteggere da attacchi che non transitano attraverso di esso. Per esempio, se venisse istaurato un collegamento di tipo dial-up da un terminale presente nella rete interna verso l'esterno, il firewall non sarà in grado di attuare le misure di sicurezza per le quali è stato creato dato che la connessione generata dall'utente non trasiterà più attraverso di esso e darà la possibilità ad un ipotetico intruso di utilizzarla per prendere il controllo della rete interna.

Un sistema di firewalling, inoltre, non è in grado di proteggere la rete interna da attacchi generati dalla rete stessa. Infatti il traffico rimarrà contenuto all'interno senza mia oltrepassarlo e pertanto escludendolo [1].

1.3.3 Tipologia di firewall

Un firewall in genere è composto da uno o più computer che si pongono tra le reti private e quelle esterne (es. internet) con il preciso scopo di controllare ciò che transita da una rete verso l'altra.

In particolare sono utilizzati per stabilire a quali servizi (www, ftp, mail,...) presenti nella rete internet è possibile accedere dalla rete locale, e quali possono essere resi disponibili alle reti esterne.



La necessità di questi controlli è dovuta al fatto che i sistemi operativi (Windows, Mac, Linux, ...), i protocolli utilizzati per lo scambio dei dati tra le reti (TCP/IP), e i programmi, hanno una serie di limiti, errori di programmazione che possono esporre la propria rete a diversi attacchi.

Questi attacchi possono permettere ad una persona con le giuste informazioni e con particolari programmi (exploit) di individuare eventuali vulnerabilità e di accedere e modificare una macchina. L'intruso potrebbe quindi apportare delle modifiche che gli consentano di accedere in qualsiasi momento senza essere scoperto (backdoor), oppure di sfruttarla per sottrarre informazioni (progetti confidenziali, password, ...), per coinvolgerla in attacchi DoS (*Denial of Service*), DDoS (*Distribuited Denial of Service*) ...

Le soluzioni adottate per proteggere le reti, viste anche le diversità dei sistemi, sono varie. L'ICSA (www.icsalabs.com/), ente che certifica l'affidabilità di un prodotto firewall, non è legato ad alcun produttore specifico di firewall ed esegue i test acquisendo i prodotti direttamente dal mercato e provandoli all'interno delle proprie strutture. Il protocollo di analisi è rigoroso e solo i prodotti che risultano positivi a tutti i test divengono certificati e acquisiscono il diritto di esporre il marchio "ICSA Labs Certified" nella documentazione tecnica, nella scatola e in tutta la comunicazione commerciale. Tale ente, ha individuato tre tipologie di firewall:

- **packet filtering;**
- **application gateway;**
- **packet inspection.**

1.3.3.1 Packet filtering

La soluzione più semplice di firewall è rappresentata da router, posti tra le due reti, che filtrano i pacchetti. Lo scopo principale di questa soluzione è, infatti, quello di intercettare i pacchetti e consentire il passaggio esclusivamente al traffico autorizzato.

Il firewall in questo caso esamina le informazioni del pacchetto dei dati, ossia rispettivamente gli indirizzi IP (di partenza e di destinazione) e i numeri di entrambe le porte contenuti nell'header (testata del messaggio). La sua analisi non va in ogni caso oltre l'header. I pacchetti vengono inoltrati o rifiutati in base alle regole di filtro impostate dall'amministratore di rete. Il problema principale con la tecnica del packet filtering è che si tratta di un sistema basato sugli indirizzi IP, elementi che offrono labili garanzie di sicurezza.

Un host è in grado di aggirare questo tipo di firewall sostituendo il proprio indirizzo IP di origine con uno autorizzato e accedere indisturbato alla rete privata. Con la tecnica del packet filtering, la gestione di alcuni protocolli risulta piuttosto critica perché obbliga all'apertura di un varco nel firewall che, se non correttamente impostato, potrebbe mettere a repentaglio la sicurezza dell'intera rete. Usare unicamente la tecnologia di filtro dei pacchetti, in linea di massima, non garantisce un livello adeguato di sicurezza. Le soluzioni oggi presenti sul mercato tendono, infatti, a fornire una combinazione di questa tecnica con altre, come per esempio quella che si basa su di un server proxy.

1.3.3.2 Application gateway

Un gateway di applicazione intercetta il traffico e autentica gli utenti a livello di applicazioni TCP/IP. Un utente sulla rete privata accede al proxy che lo autentica, dopodichè ha accesso al server remoto che si trova su internet. In maniera analoga, tutte le comunicazioni provenienti da internet e dirette alla rete privata vengono ricevute dal proxy, analizzate e successivamente, previa autorizzazione, inoltrate a destinazione. Lavorando a questo livello, è necessario predisporre un server proxy per ogni applicazione. Per evitare accessi indesiderati, il sistema di autenticazione deve essere assolutamente sicuro.

1.3.3.3 Packet inspection

L'approccio di un firewall che si basa sulla tecnica della packet inspection è rappresentato dall'analisi dei pacchetti piuttosto che dal solo filtraggio: in altri termini vengono controllati anche i contenuti del pacchetto IP oltre agli indirizzi. Questa tipologia di firewall, detta anche *circuit-level gateway*, offre il massimo grado di affidabilità. A differenza dell'application gateway, questo tipo di soluzione richiede, al momento dell'accesso da parte di un utente al server internet, la modifica del software client. La nuova generazione di browser, per esempio, ormai supporta automaticamente questo tipo di accesso [2].

1.4 Perché usare i firewall

Il termine firewall usato per descrivere un filtro di rete deriva dal nome con cui vengono chiamate quelle barriere fisiche separatrici che impediscono al fuoco di diffondersi da un'area all'altra, usate ad esempio dai vigili del fuoco. I filtri di rete vengono chiamati firewall perché il loro compito è similare: impedire, entro certi limiti, gli attacchi di malintenzionati che possono carpire o distruggere i dati o provocare attacchi tipo di rifiuto del servizio (*Denial of Service*, DoS), oppure

cercare di sovraccaricare le risorse di elaborazione (riempire il disco, intasare la connessione internet, ecc).

Occorre subito precisare che non è sufficiente avere un firewall, anche se ben configurato, per risolvere tutti i problemi di sicurezza, e né si può pensare che installare un firewall renda inutili i backup di sicurezza dei dati, che conviene sempre programmare opportunamente, indipendentemente da quanto sia ristretta la politica di sicurezza adottata, perché rappresentano l'unico modo di tutelarsi da guasti o distruzione dell'hardware. Similmente avere un firewall non basta a risolvere il problema dello spam, che sussiste tutte le volte che la rete consente la ricezione delle e-mail, o quello dei virus, per cui si rende ancora necessario usare del software anti-virus e tenerlo aggiornato per una migliore protezione. Il più alto livello di sicurezza consiste nell'impedire qualsiasi tipo di traffico di rete, bloccandolo completamente (o equivalentemente staccando la spina del cavo di rete). Ovviamente questo non è praticabile perché sarebbe inutile avere una connessione ad internet. Nel configurare un firewall si può partire con una regola che blocca tutto e poi inserire man mano le regole che servono a far passare tutto e solo il traffico che si desidera passi. Questo è un buon metodo per essere quanto più possibili restrittivi.

Nonostante i suoi limiti, il firewall può essere una componente molto importante per controllare l'utilizzo di una rete forzando una politica di sicurezza prestabilita.

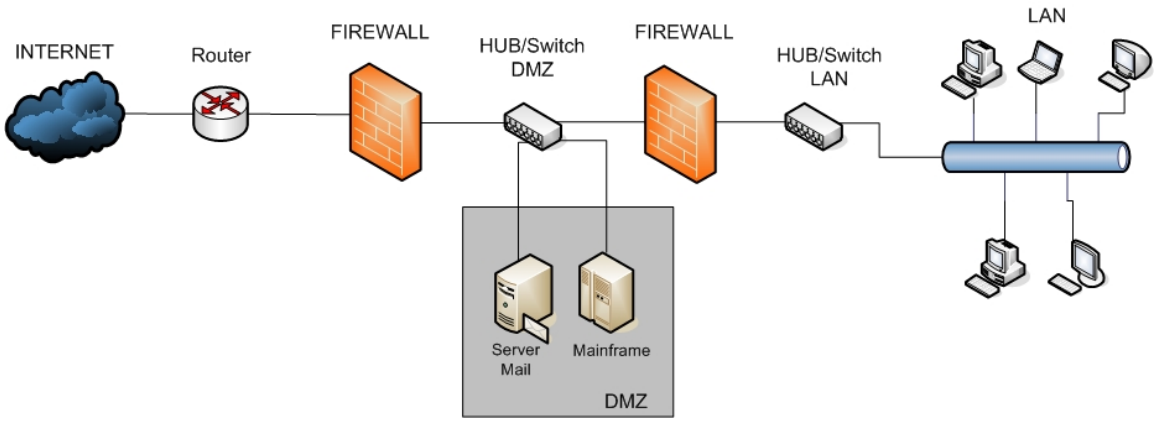
Come caso limite, invece di un'intera rete, supponiamo di avere un unico server di cui si è amministratori e responsabili e che naturalmente si vuole proteggere dalle intrusioni. Supponiamo che su questo server siano stati configurati opportunamente i servizi di rete, restringendo l'accesso quando applicabile, usando quando più possibile protocolli sicuri (es. ssh al posto di telnet e ftp) e che l'amministratore ci lavori e stia piuttosto attento a mantenere il software aggiornato e a prendere tutte le altre misure di controllo del sistema.

Potreste pensare in tal caso di non aver bisogno di un firewall. Eppure potrebbe essere lo stesso utile configurare un firewall (meglio ancora se hardware) a protezione della macchina anche in questo caso in cui la macchina è gestita da un esperto di sicurezza informatica, perché il firewall garantisce un ulteriore livello di sicurezza del sistema, che i malintenzionati devono scavalcare. La sicurezza non è mai troppa: sempre meglio avere più livelli di sicurezza che uno soltanto.

Viceversa per proteggere le macchine di una rete usate da utenti inesperti di sicurezza informatica, un firewall può rivelarsi molto utile. Se si tratta di una rete con molti utenti, e quindi molti computer che condividono una o più connessioni ad internet a banda larga (ad es. una o più linee T1 o T3), sarà sicuramente impossibile riuscire a gestire la sicurezza di ogni singolo calcolatore, anche per il semplice fatto che avete troppi calcolatori da gestire. Alcuni calcolatori potrebbero essere portatili di eventuali ospiti o di proprietà dei dipendenti e quindi su questi non si ha alcun controllo. Utenti inesperti potrebbero attivare programmi server sui loro calcolatori come server Web, FTP di posta elettronica o altro che sono facilmente vulnerabili perchè non correttamente configurati, oppure sono versioni che presentato bachi ben noti e facilmente sfruttabili dai malintenzionati. Alcuni sistemi Windows per uso client di qualche anno fa (95, 98) sono molto poco sicuri, in quanto non hanno meccanismi di controllo di ciò che un utente può fare (i cosiddetti “permessi di utente”): sono sostanzialmente sistemi monoutente in cui si lavora sempre con i permessi dell’amministratore. Qui i virus e i worm possono fare grossi danni. Le nuove versioni (2000, XP) sono più sicure, specie se usate come utente non privilegiato per tutto il lavoro quotidiano e se solo voi o collaboratori di cui vi potete fidare conoscono le password di amministratore. In ogni caso comunque chi ha l’accesso fisico alla macchina potrebbe sostituire il sistema operativo.

In questa situazione descritta, senza un firewall che protegga l’intera rete (un firewall posto di guardia ad ogni punto di accesso verso l’esterno), un

malintenzionato esterno potrebbe facilmente guadagnare l'accesso ad una delle macchine interne ed usarla come base per penetrare su ancora altre macchine della rete interna (per esempio sui server che gestiscono i servizi e le informazioni più importanti e li mettono maggiormente a disposizione delle macchine interne anziché di quelle esterne) o di altre reti esterne, usando, la rete attaccata, come ponte per una azione criminale ai danni di terzi (si rischia di essere ingiustamente accusati). In questa situazione avere un firewall ben configurato e gestito rappresenta di sicuro una necessità irrinunciabile volendo garantire un pur minimo livello di sicurezza. Il firewall dovrebbe consentire solo a quei calcolatori gestiti da esperti della sicurezza di fornire servizi di rete verso l'esterno. Nel contempo occorre venire incontro alle esigenze degli utenti, cercando di bilanciare quanto meglio possibile sicurezza e funzionalità della vostra rete. Ad esempio gli utenti vorrebbero poter essere in grado di scaricare la posta elettronica da casa o collegarsi al file server per poter lavorare. Si potrebbe anche scegliere di porre dietro un firewall tutti i client e i server che devono fornire servizi solo per la propria rete interna, mentre mettere a monte del firewall, ossia tra il firewall e la connessione ad internet (nella cosiddetta area demilitarizzata, o *DMZ Delimitarized Zone*), quei server che devono fornire servizi verso l'esterno (opportunamente configurati e controllati). In questo modo si evita l'overhead del packet filtering per questi server.



I campi di applicazione dei firewall sono i più svariati, dalle piccole alle

grosse reti, sia che si tratta di reti connesse ad internet oppure di reti solo private. Ad esempio un firewall è utile anche per proteggere una piccola rete domestica quando ci si collega ad internet, sia che il collegamento avvenga saltuariamente tramite modem analogico e normale linea telefonica, sia se si ha una connessione a banda larga e permanente, come DSL o tramite un modem per cavo. Naturalmente una connessione, permanente, o usata per molte ore al giorno, è più soggetta ad essere oggetto di eventuali attacchi di malintenzionati.

1.5 Cosa può controllare un firewall

Un firewall, che può essere sia un dispositivo hardware, oppure può essere implementato via software, può filtrare e controllare le informazioni che provengono da internet verso la propria rete privata o singolo computer. Può anche filtrare e controllare quelle che escono dalla propria rete privata verso internet, quindi in generale un firewall può filtrare in entrambi i sensi e il filtraggio riguarda sia i pacchetti in uscita che in ingresso. Il packet filtering si attua specificando una serie di regole (*rules*) che rappresentano la configurazione del firewall: quando un pacchetto fa corrispondenza con una regola può essere scartato o inoltrato a seconda di quello che stabilisce la regola stessa.

Oltre al packet filtering, esiste un nuovo metodo di filtraggio più fine e sofisticato, detto *stateful inspection*, che può essere combinato insieme al packet filtering. Questo metodo consente di generare dinamicamente delle regole di filtraggio di tipo packet filter, che hanno un certo tempo di scadenza anziché essere statiche. Il principio è semplice: il firewall lascia passare alcuni pacchetti di richiesta provenienti dall'interno (essendo stato abilitato a fare ciò), nel contempo salva in un database alcune informazioni fondamentali riguardanti queste richieste. I pacchetti in entrata vengono poi confrontati con queste informazioni chiave, in modo che vengano accettati solo pacchetti in ingresso che rappresentano delle risposte alla richieste inviate dagli utenti, scartando quindi le pseudo-risposte non valide.

Spesso le funzionalità di un proxy server e quelle di un firewall vengono integrate in un unico software; i proxy sono calcolatori che recuperano le informazioni da una connessione internet in vece dei loro client a cui poi le inviano (e viceversa i client fanno al proxy le richieste esterne e questo le rifà agli host sparsi su internet in loro vece), provvedendo una cache centralizzata per ridurre il carico sulle connessioni di rete verso l'esterno e aumentare la velocità di navigazione e una qualche forma di controllo di accesso (firewall più o meno evoluto incorporato nel proxy). Inoltre un proxy può fornire procedure di autenticazione più sicure di quelle di protocolli quali telnet e ftp che fanno passare le password in chiaro.

Si può persino impedire l'accesso a siti offensivi, controllandone il contenuto e confrontandolo con una lista di parole chiave. Naturalmente in questo modo si può bloccare anche del contenuto legittimo contenente una parola chiave. Effettuare il blocco in base agli indirizzi IP dei siti vietati non è raccomandabile in quanto questi possono venire riassegnati ad enti diversi ed è difficile tenerne aggiornata la lista che sarà probabilmente molto lunga, rischiando così di bloccare dei contenuti ritenuti leciti e facendo passare invece buona parte dei contenuti da ritenersi invece illeciti. Effettuare il blocco in base ai nomi host richiede la risoluzione dei nomi tramite il protocollo dei DNS. Un firewall molto efficiente posto a protezione di un'intera rete solitamente non opera a questo livello, mentre un firewall software potrebbe avere questa feature.

Un firewall hardware è certamente più sicuro di uno software e comunque non è molto costoso. Molti router posseggono funzionalità di filtraggio, combinando così le funzioni di un router e di un firewall in un unico componente.

Un router è una macchina che effettua il forwading dei pacchetti tra due o più reti, che possono adottare anche protocolli diversi, per cui il router dovrà essere in grado di effettuare una conversione nel formato dei dati. I router sono un componente fondamentale di internet, senza di questi internet non potrebbe

esistere come rete mondiale; infatti la rete internet è la confederazione di molte reti, e i router sono quei dispositivi che consentono a tutte queste reti di comunicare tra loro. I router sono spesso dispositivi hardware dedicati, amministrabili tramite software opportuno, ma un qualsiasi host con software opportuno che abbia almeno una connessione per ciascuna rete tra cui si vuole effettuare il routing, può agire da router software [3].

1.6 Quello che i firewall non possono fare

Sebbene i firewall siano una parte utile di un programma di sicurezza della rete, non sono una panacea. Se gestiti in modo appropriato sono utili, ma non faranno ogni cosa. Se usati in modo inappropriato, l'unica cosa che procurano è un falso senso di sicurezza.

I firewall sono inutili contro attacchi dall'interno. Un attacco interno può provenire da un utente legittimato che è passato all'altra sponda, o da qualcuno che ha ottenuto l'accesso a una macchina interna tramite altri mezzi. Anche il codice ostile che viene eseguito su una macchina interna, magari arrivato tramite un virus dì posta elettronica o sfruttando un buffer overflow sulla macchina, può essere visto come un attacco interno.

Alcune organizzazioni hanno modelli di minaccia interna più seri di altre. Alcune banche dispongono di dipartimenti legali in piena regola che spesso controllano le proprie reti interne con molta attenzione, e smontano le macchine quando sospettano qualcosa. Il loro scopo è vedere quali danni sono stati fatti. Le organizzazioni militari hanno pericoli interni altrettanto elevati. (Vi sono statistiche, spesso citate, sulla percentuale di attacchi proviene dall'interno. La metodologia che sta dietro queste indagini è così carente che non è possibile credere a nessuna di queste cifre. Tuttavia, è sicuri che gli interni rappresentino notevoli minacce).

Se il firewall è l'unico meccanismo di sicurezza di cui si dispone, e qualcuno riesce a introdursi tramite qualche altro meccanismo, il sistema è nei guai. Per esempio, se si esegue il rilevamento dei virus solo sul gateway della posta elettronica, la sicurezza può andare in fumo se qualcuno inserisce un dischetto infettato o scarica un eseguibile dal Web. Qualsiasi connessione backdoor che aggira il filtraggio può dimostrare l'efficacia limitata dei firewall. I problemi di gestione di MIME, come i buffer overflow, hanno causato problemi di sicurezza che esulano dall'ambito di gestione per cui sono stati concepiti i firewall.

Il concetto di un guscio solido che racchiude un ripieno morbido dà sicurezza solo se non esiste alcun modo per arrivare all'interno. Oggi, questo potrebbe essere irrealistico.

La mancanza di cooperazione interna è un caso speciale dell'attacco interno, ma fondamentalmente è un problema legato alle persone. Per gli utenti che non vogliono cooperare è facile impostare tunnel, come IP sopra HTTP. A quel punto, il filtraggio IP al livello IP più basso è inutile.

I firewall agiscono a un ben preciso livello dello stack dei protocolli, il che significa che non guardano nulla ai livelli più alti. Se si sta eseguendo un filtraggio dei numeri di porta solo al livello di trasporto, non si vedranno i problemi a livello di SMTP. Se si esegue il filtraggio a livello di SMTP, si potrebbero saltare i problemi creati dai dati nelle intestazioni di posta; se si guardano le intestazioni, si potrebbero perdere virus e cavalli di Troia. È importante determinare i rischi a ogni livello e agire di conseguenza. Vi sono comunque dei compromessi che sarà necessario accettare. Il filtraggio a livello più alto è più intrusivo, più lento in termini di elaborazione e meno completo, poiché man mano che si sale nello stack aumentano notevolmente le opzioni di elaborazione per ciascun pacchetto.

La scansione dei virus di posta elettronica sembra essere vincente per i siti

Windows. Se non altro, buttare via tutta la posta elettronica infettata in corrispondenza del gateway può risparmiare parecchia larghezza di banda. (Ma una buona strategia consiste nell'eseguire un tipo di scanner di virus nel gateway, e un altro sui desktop. Il software AV non è perfetto). Al contrario, cercare di scansire i download FTP è inutile per la maggior parte dei siti. Le trasformazioni dei dati, come la compressione, rendono l'operazione praticamente impossibile, soprattutto in relazione alla velocità della linea. Decidere dove e quanto filtrare è un problema di rapporto tra rischio e costi. Vi è sempre un livello più alto, comprese le persone che eseguono stupide istruzioni nella posta elettronica, e che non sono così facili da filtrare.

Un altro problema legato ai firewall è quello della fiducia transitiva, presente. Se A si fida di B attraverso il suo firewall, e B si fida di C, volente o nolente (e sempre ammesso che lo sappia) A si fida di C.

Infine, i firewall possono contenere degli errori, o non funzionare secondo le aspettative. La migliore amministrazione non può fare nulla per opporsi a un firewall che non funziona come dovrebbe [4].

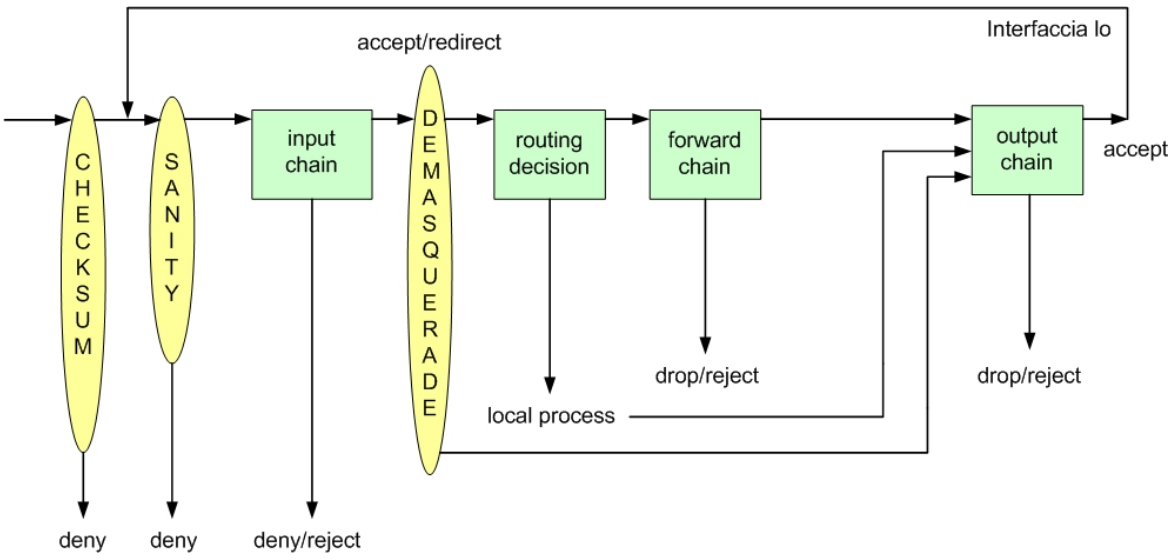
CAPITOLO 2

IMPLEMENTAZIONE DI TRE SISTEMI FIREWALL

2.0 Struttura e funzionalità principali di IPTables

IPTables è un pacchetto open source operante in ambiente GNU/Linux che utilizza gli strumenti messi a disposizione dai kernel della serie 2.4 e successivi.

Il programma IPTables rappresenta solo un'applicazione in spazio utente. Quando un pacchetto IP entra nel firewall, viene passato al corrispondente driver all'interno del kernel. Quindi il pacchetto attraverserà una serie di stati prima di essere inviato ad un'applicazione locale o inoltrato attraverso un'altra interfaccia di rete.



IPTables è strutturato internamente attraverso delle catene (chains). Un pacchetto, una volta immesso in una di queste, può essere bloccato o accettato, a seconda delle regole impostate dall'utente. Un altro concetto importante è quello di tabella (table). Non appena IPTables viene caricato all'interno del kernel, verranno create tre tabelle e diverse catene già preimpostate [6].

2.1 Le catene e le regole

La configurazione di un firewall avviene attraverso la dichiarazione di regole (rules). Ogni regola dovrà essere inserita in un contesto più ampio, ossia all'interno di una catena (chain).

Una regola può essere definita come un'indicazione alla quale il firewall dovrà attenersi per decidere se accettare o rifiutare i pacchetti appartenenti alle differenti connessioni. In ultima analisi, consiste nell'intraprendere una particolare azione nel caso in cui si verifichino alcune precondizioni. Per ogni pacchetto e per ogni regola, il kernel, verificherà che tutte le condizioni siano rispettate (match) e, in tal caso, inoltrerà il pacchetto verso la catena specificata (operazione di jump).

Le catene principali, ordinate a partire dall'arrivo del pacchetto fino alla sua successiva ripartenza attraverso un'altra interfaccia di rete, sono:

- PREROUTING: ogni pacchetto che giunge al firewall proveniente da qualsiasi interfaccia, arriva in questa catena, a monte della decisione di routing. Viene quindi principalmente utilizzata per il DNAT. Infatti, la modifica dell'indirizzo di destinazione, influenzerà direttamente la decisione di routing da prendere per quanto riguarda il pacchetto;

- INPUT: ogni pacchetto destinato direttamente al firewall entra in questa catena. In tal modo sarà possibile definire a quali servizi presenti sulla macchina permettere l'accesso dalle diverse interfacce;

- OUTPUT: ogni pacchetto che viene generato direttamente dal firewall transita da questa catena;

- FORWARD: ogni pacchetto che viene generato da un'interfaccia di rete destinato ad un'altra entra in questa catena. Sarà quindi possibile

gestire, a seconda delle esigenze della propria rete, le diverse interconnessioni permettendo, ad esempio, alla LAN interna di accedere ad internet ma impedendo l'accesso dall'esterno alla propria rete privata;

- POSTROUTING: ogni pacchetto arriva in questa catena a valle della decisione di routing, appena prima di lasciare il firewall. Viene principalmente utilizzata per il SNAT. Infatti, la modifica dell'indirizzo sorgente, appena prima che il pacchetto lasci il firewall, permette la sostituzione dell'IP di rete privata, quindi non indirizzabile da internet, con un altro indirizzo pubblico (solitamente quello del firewall stesso).

Ogni regola permette di ricercare dei riscontri su un'ampia gamma di parametri; tra i più comuni ed utilizzati si devono citare la verifica del protocollo, l'indirizzo di sorgente e di destinazione, le porte coinvolte nella connessione. Tutti questi elementi possono essere utilizzati per effettuare un corretto filtraggio e per prendere una decisione per quanto riguarda la sorte del pacchetto [6].

Per capire quando e come agiscano queste catene seguiamo il percorso di un pacchetto IP dalla scheda di rete esterna a quella interna senza la presenza del software di firewall IPTables. Dopo essere entrato dalla scheda di rete esterna, il pacchetto viene aperto e ne viene analizzato l'header alla ricerca dell'indirizzo di destinazione. Questo indirizzo viene confrontato con la tabella di routing della macchina e quindi instradato verso una porta locale o verso la scheda di rete appropriata se l'indirizzo di destinazione è differente da quello associato al firewall.

Prima di proseguire, visto che abbiamo tirato in ballo la tabella di routing, vediamo che funzione svolge nel processo di trasporto del pacchetto.

La tabella di routing, viene utilizzata per decidere dove instradare il pacchetto IP in base all'indirizzo di destinazione presente nell'header.

Essa contiene un'associazione tra blocchi di indirizzi internet e risorse con cui tali indirizzi possono essere raggiunti. Le risorse possono essere interfacce di rete locali o indirizzi IP di computer detti "gateway".

Di seguito vi è un esempio di una banale tabella di routing di una machina con una sola interfaccia ethernet. Linux nomina le interfacce di rete ethernet con il suffisso `eth`, numerandole poi in base alla posizione all'interno degli slot ISA/PCI.

Kernel IP routing table

)estination	Gateway	Genmask	Flags	Metric	Ref	Use	Iface
127.0.0.0	*	255.0.0.0	U	0	0	0	lo
192.168.0.0	*	255.255.255.0	U	0	0	0	eth0
0.0.0.0	192.168.0.1	0.0.0.0	UG	0	0	0	eth0

Dalla tabella sovrastante si evidenzia:

- l'indirizzo `127.0.0.1`, il così detto indirizzo di loopback, mappato sull'interfaccia (virtuale) `lo`;
- gli indirizzi della rete locale da `192.168.0.0` a `192.168.0.255` mappati sulla scheda di rete `eth0`;
- l'indirizzo `0.0.0.0/0.0.0.0`, l'ultima risorsa nel caso che le altre rotte non vengano applicate al pacchetto e che, di solito, corrisponde all'indirizzo del router/gateway verso internet.

Ritorniamo all'analisi del tragitto percorso dal nostro pacchetto IP e vediamo cosa accade in presenza del firewall IPTables.

Il pacchetto entra dall'interfaccia esterna e viene sottoposto, prima del processo di routing, all'applicazione delle direttive presenti nella lista

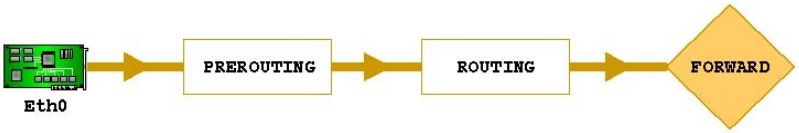
PREROUTING.

Usualmente, in tale catena vengono inserite regole che tendono a evidenziare il pacchetto per distinguerlo dagli altri pacchetti ed eseguire su di esso adeguate operazioni nelle successive fasi del processo di trasporto.

In tale fase vengono anche applicate le regole per la gestione del destination NAT (DNAT).

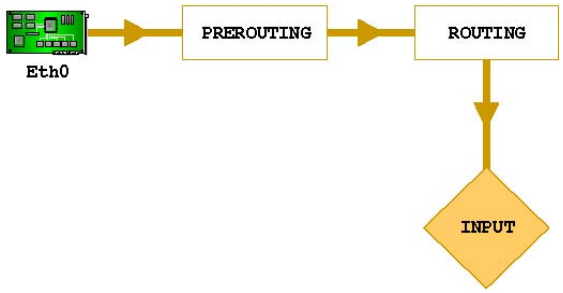
Il pacchetto subisce il processo usuale di routing in base alla tabella presente nella macchina locale.

Se il pacchetto, in base alla tabella di routing, è destinato all'interfaccia di rete interna vengono applicate le regole descritte nella lista di FORWARD.



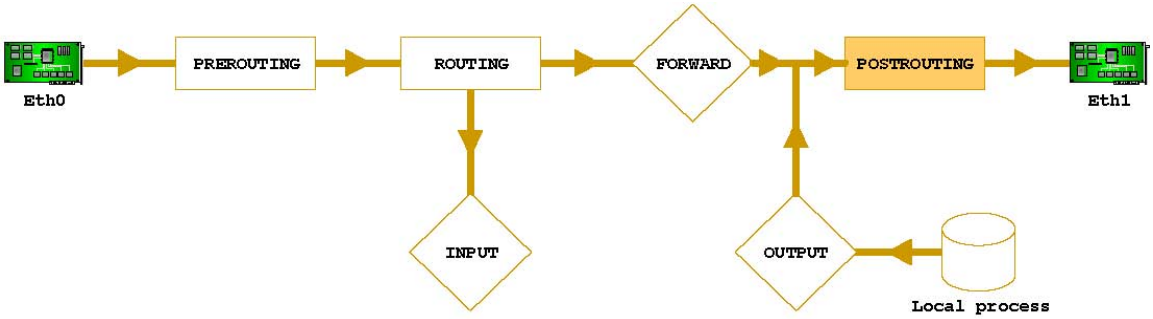
Usualmente sono questi i filtri più importanti in quanto definiscono cosa può passare dall'esterno verso l'interno e cosa no.

Se il pacchetto è destinato, in base alla tabella di routing, alla macchina locale vengono applicate le regole descritte nella lista di INPUT. Questi filtri proteggono il firewall stesso da accessi indesiderati.



Se il pacchetto ha come sorgente la macchina locale, ossia è stato generato da un processo della macchina locale vengono applicate, al pacchetto, le regole di OUTPUT.

Sia nel caso di forward che di output, prima di uscire dalla scheda di rete interna, il pacchetto subisce l'applicazione delle direttive di POSTRUTING. In tale fase vengono di solito applicate le regole per il source NAT (SNAT).



In ognuno di questi step ogni direttiva si chiede sostanzialmente: "se l'header del pacchetto verifica certe condizioni, che cosa devo fare del pacchetto" ? La risposta a questa domanda può essere o di accettare il pacchetto che continua nel suo percorso all'interno delle altre direttive e degli altri step o rigettare il pacchetto che viene definitivamente buttato via.

In ogni step, se il pacchetto non verifica nessuna delle condizioni impostate,

può essere definita una regola di default da applicare al pacchetto che verrà quindi accettato o rigettato.

Usualmente si ritengono sicuri ed accettabili i pacchetti provenienti dall'interno e destinati verso l'esterno e quindi, in particolare, i pacchetti che hanno come sorgente il firewall saranno permessi e quindi l'impostazione ovvia di default del processo di OUTPUT sarà quella di ACCEPT.

Quello che si vuole invece evitare, a meno di eccezioni, e che dall'esterno si possa impunemente accedere verso l'interno. Ecco perché è usuale impostare a DROP la configurazione di default dei processi di FORWARD e di INPUT.

Sarebbe un errore lasciare ad ACCEPT queste regole e tentare di chiudere tutti i servizi interni. Il firewall serve, in particolare, per chiudere tutto, anche quello che neanche si sa di avere aperto. Negando tutto e accettando solo ciò che si considera come corretto si è sicuri che se dall'esterno di accede ad una data risorsa interna è perché siamo stati noi a richiederlo.

Il primo banale script di configurazione del nostro firewall sarà quindi (il flag `-P` imposta per l'appunto la politica di default per il processo) [5]:

```
-P INPUT DROP

-P FORWARD DROP

-P OUTPUT ACCEPT
```

2.2 Le tabelle

Innanzitutto analizziamo in dettaglio le tabelle che vengono create:

2.2.1 FILTER

Si tratta della tabella più importante e viene utilizzata per effettuare il filtraggio vero e proprio sui pacchetti che transitano attraverso le interfacce del firewall (packet filtering).

2.2.2 NAT

Questa tabella deve essere utilizzata per eseguire NAT (Network Address Translation), quindi per tradurre l'indirizzo sorgente o destinazione presente nell'header dei pacchetti IP). IPTables implementa due tipi diversi di NAT:

- SNAT (Source Network Address Translation): Tecnicamente l'SNAT modifica, nell'header dei pacchetti, l'indirizzo IP del sorgente facendo credere al destinatario del pacchetto che esso provenga da un altro indirizzo IP.

Questo permette anche a chi non è fisicamente in internet di navigare per la rete. Spieghiamo questo con un esempio:

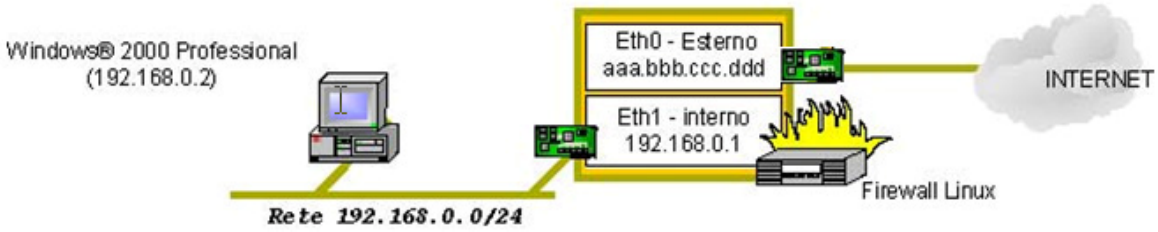
Consideriamo due computer, uno con Windows 2000 Professional e uno con Linux, collegati alla stessa rete locale.

Ai due computer sono assegnati due indirizzi IP intranet del blocco `192.168.0.0/24` e, in particolare, la macchina Linux ha un indirizzo IP assegnato alla sua scheda di rete `eth1 192.168.0.1` mentre la macchina Windows ha l'indirizzo `192.168.0.2`.

Il protocollo TCP/IP è indipendente dal sistema operativo e quindi, tramite

esso, le due macchine “si vedono” sulla rete locale.

Sulla macchina Linux vi è anche un’altra scheda di rete `eth0` alla quale viene assegnato un indirizzo internet `aaa.bbb.ccc.ddd` dal DHCP. La macchina Linux ha quindi, due indirizzi IP assegnati, uno interno, con cui vede la macchina Windows, e uno esterno, tramite il quale risulta visibile e vede la internet.



La prima cosa da fare, per far accedere ad internet anche la macchina Windows, consiste nell’informare il modulo TCP/IP della macchina, che esiste sulla rete locale, una macchina che sa come inviare pacchetti ad internet.

Ciò consiste nel configurare il gateway di default della macchina Windows impostandolo con l’indirizzo IP della macchina Linux.

Per tale operazione viene utilizzato l’indirizzo locale `192.168.0.1` in quanto l’altro è già un indirizzo internet e la macchina Windows 2000 non sa come raggiungere la rete.

Con questa configurazione una richiesta della macchina Windows destinata a internet giunge sulla macchina Linux che instrada il pacchetto sulla sua rotta di default che, risulta settata proprio su tale linea.

Sembra che già tutto funzioni ma in realtà non funziona niente. L’unica macchina che il router ha autorizzato a navigare è la macchina Linux. Quando il router vede arrivare dei pacchetti dalla macchina Windows con indirizzo di sorgente `192.168.0.1` rigetta tali pacchetti in quanto, dal suo punto di vista, sono pacchetti anomali perché gli unici pacchetti che dovrebbero arrivare devono avere

indirizzo di sorgente `aaa.bbb.ccc.ddd`.

Quando arriva il pacchetto dalla macchina Windows IPTables si segna l'header che dovrebbe assumere il pacchetto di risposta ad esso associato e sostituisce l'indirizzo di sorgente `192.168.0.2` del pacchetto in transito con il suo indirizzo internet `aaa.bbb.ccc.ddd` instradandolo verso la `eth0` connessa ad internet.

Ora il router non ha più nessun motivo per rigettare il pacchetto in quanto, dal suo punto di vista, esso proviene dal soggetto legittimamente autorizzato a navigare lungo quella connessione.

La cosa non finisce però qui in quanto il server remoto invia la risposta alla macchina Linux che si vede arrivare una risposta per una richiesta che non ha fatto. Qui interviene di nuovo il SNAT che si accorge, controllando l'header del pacchetto con quello salvato sopra, che questo flusso di dati remoti va dirottato alla macchina Windows.

Sostituisce allora all'indirizzo del destinatario, attualmente settato sull'header del pacchetto di risposta, l'indirizzo IP della macchina Windows e glielo invia. La macchina Windows si vede arrivare il pacchetto di risposta che si aspettava.

La macchina Windows naviga quindi per "interposta persona" in quanto è la macchina Linux che la fa navigare mascherando di volta in volta l'indirizzo sorgente della macchina locale.

Vediamo ora tecnicamente come si configurano le cose sulla macchina Linux con IPTables.

La configurazione dell'SNAT è molto semplice e consiste nell'attivazione della tabella di NAT nel processo di POSTROUTING tramite l'impostazione:

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

deve `eth0` è l'interfaccia di uscita verso internet.

Come si vede non appare alcun indirizzo IP nella configurazione in quanto il sistema provvede automaticamente a ricavare l'indirizzo dinamico assegnato di volta in volta dal provider e di utilizzarlo per lo SNAT.

Se la situazione è quella di un indirizzo statico con pochi indirizzi IP è meglio utilizzare questa configurazione:

```
-t nat –A POSTROUTING –o eth0 –j SNAT –-to xxx.xxx.xxx.xxx
```

dove `eth0` è l'interfaccia di rete lato internet e `xxx.xxx.xxx.xxx` è l'indirizzo internet di questa interfaccia.

Tale regola va applicata alla catena di POSTRUTING per far si, che almeno fino all'ultimo momento, prima di uscire dall'interfaccia verso internet, il pacchetto sia ancora identificato come proveniente dalla macchina interna e su cui poter applicare regole di filtro o di routing. Se questo mascheramento avvenisse prima, il pacchetto sembrerebbe del tutto indistinguibile da un pacchetto generato localmente dal box Linux e non potrebbero quindi essere applicati eventuali filtri personalizzati.

- DNAT (*Destination Network Address Translation*): Analogamente al source NAT anche il DNAT agisce modificando l'header dei pacchetti in transito ed in particolare l'indirizzo e la porta di destinazione.

Al contrario dell'SNAT, ma per lo stesso motivo, il DNAT viene applicato nel processo di PREROUTING così che tutti i restanti processi di IPTables e di routing agiscono sul pacchetto già modificato.

L'operazione di DNAT può essere utilizzata per operazioni di "*port forwarding*" o di "*trasparent proxy*".

Ciò che ci interessa è il "*port forwarding*" che è quell'operazione per cui una porta presente su di un server in realtà è un "puntatore" ad un servizio presente su di un'altra porta molto spesso attiva su di un altro server. Tutte le richieste di connessione sulla prima porta vengono inoltrate al servizio reale attivo sulla seconda porta.

Un utilizzo del "*port forwarding*" si ha nella situazione in cui uno o più servizi presenti in una rete con indirizzi intranet debbono essere esportati e visibili anche sulla rete internet.

Portiamo quindi ad esempio una azienda che ha a disposizione quattro indirizzi IP nell'intervallo da `194.244.12.0` a `194.244.12.3` con subnet `255.255.255.0`.

Dei quattro indirizzi, lo zero è l'indirizzo della rete, l'uno è l'indirizzo del router che collega l'azienda al provider e il tre è l'indirizzo di broadcast. L'unico indirizzo libero risulta quindi `194.244.12.2` e visto che l'azienda in realtà possiede molti computer che desiderano navigare sulla Rete, ha utilizzato l'SNAT attivandolo su di una macchina Linux. L'indirizzo libero è stato assegnato all'interfaccia di rete esterna mentre a quella interna è stata data un indirizzo intranet della classe `192.168.0.0/24`.

L'azienda decide quindi di creare un proprio sito, e lo posiziona all'interno, della rete aziendale per averne una migliore gestione e manutenzione.

Attivare il server web sul firewall potrebbe essere impossibile per incompatibilità di sistema operativo ma, anche nel caso in cui non esistesse questa incompatibilità, l'installazione di ulteriori servizi sul firewall è totalmente inaccettabile. Il firewall, come unico baluardo tra noi è "il nemico" deve essere

totalmente sicuro e ogni servizio aggiuntivo tenderebbe a ridurre questo fattore di sicurezza.

Il server viene quindi agganciato alla rete interna e gli viene assegnato l'indirizzo intranet `192.168.0.1`.

Se è vero che l'installazione del servizio web sul firewall è inaccettabile è però vero che tentare di mappare la porta web del firewall con la porta web del server `192.168.0.1` non preclude assolutamente la sicurezza della rete.

L'idea è proprio questa: esternamente il server web verrà visto attivo sulla porta `80` (web) dell'indirizzo `194.222.12.2` ma tutti i pacchetti diretti su questa posta verranno poi dirottati dal firewall sulla porta `80` del server `192.168.0.1`.

Seguiamo un ipotetico pacchetto TCP/IP di richiesta di una pagina dal sito e il relativo pacchetto di risposta, per comprendere il funzionamento.

Il pacchetto di richiesta diretto verso la porta `80` del server `194.222.12.2` entra dalla scheda esterna `eth0` del firewall.

Nel processo di PREROUTING il firewall verifica che questo pacchetto corrisponde ad una determinata regola e ne modifica il destinatario (DNAT) sostituendo l'indirizzo di destinazione con il `192.168.0.2`.

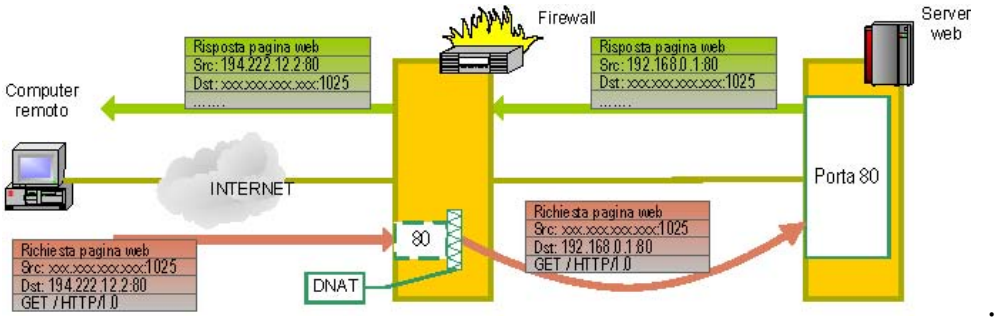
La tabella di routing vede ora un pacchetto destinato a questo host interno e lo instrada sulla scheda `eth1` permettendo al pacchetto di giungere sul server web realmente presente sulla rete.

Il server web interno processa la richiesta e prepara un pacchetto di risposta.

Questo pacchetto torna quindi sul firewall che riconosce questo come pacchetto di risposta relativo al pacchetto che poco prima lui aveva modificato; lo modifica sostituendo all'indirizzo di sorgente intranet il proprio indirizzo e lo

inoltra all'host remoto.

Risultato, dal punto di vista dell'host remoto:



La configurazione di IPTables per ottenere questo risultato è molto semplice:

```
-t nat –A PREROUTING –p tcp –d 194.222.12.2 –-dport 80 –j DNAT

–-to 192.168.0.1
```

Come si vede, non è stato necessario specificare la porta di destinazione in quanto la mappatura delle porte è la stessa. Se internamente il server web era attivato su una porta non standard e quindi differente dalla porta `80` si sarebbe dovuto aggiungere questa informazione alla riga di configurazione del DNAT.

Ovviamente quello che funziona per un servizio funziona anche con molti. Si possono così avere più server interni e mapparli esternamente utilizzando il solo indirizzo IP del firewall. Se però si cerca di esportare due server dello stesso tipo, per esempio due server web attivati su due macchine differenti, solo uno dei due potrà essere visto esternamente presente su firewall sulla porta web standard. L'altro dovrà essere mappato su una porta `xx` non standard e sarà visibile esternamente solo tramite l'url [5].

2.2.3 MANGLE

Impostando opportune regole in questa tabella sarà possibile modificare internamente il pacchetto che vi transita. Risultano possibili infatti modifiche ai valori ToS (*Type of Service*) e TTL (*Time to Live*) dell'header. Si può inoltre imprimere una marcatura al pacchetto per destinarlo ad un trattamento successivo attraverso appositi programmi esterni. Ad esempio, in alcune situazioni, potrebbe essere conveniente stabilire una diversa politica di routing basata sulla marcatura dei pacchetti o implementare un particolare sistema di QoS (*Quality of Services*). Tutto ciò è possibile utilizzando appositi strumenti messi a disposizione, per esempio, dal pacchetto iproute2.

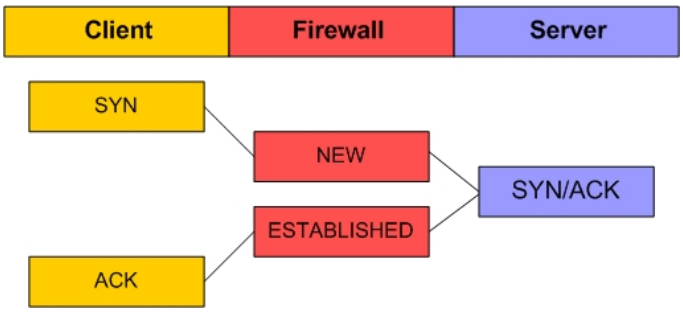
2.3 La macchina a stati

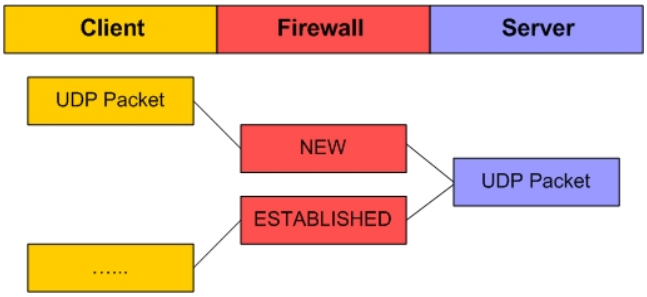
Un concetto molto importante e di estrema utilità che viene implementato da IPTables è quello di macchina a stati. Si tratta di una macchina virtuale che realizza un sistema di tracciamento delle connessioni (*connection tracking*). Sono previsti quattro stati:

- NEW La macchina stabilisce che una connessione si trova in questo stato nel caso in cui un pacchetto fosse il primo relativo ad una particolare connessione. Per esempio, nel caso del protocollo TCP, la presenza del bit SYN indica esplicitamente l'inizializzazione di una nuova connessione;

- ESTABLISHED La macchina entra in questo stato quando viene rilevato del traffico in entrambe le direzioni. Nel caso del protocollo TCP, una connessione viene definita ESTABLISHED non appena viene riscontrato un pacchetto di ritorno con i bit SYN e ACK attivi (termine del *three-way handshake*); nel caso di UDP, invece, quando viene ricevuto un pacchetto di risposta ad uno precedentemente

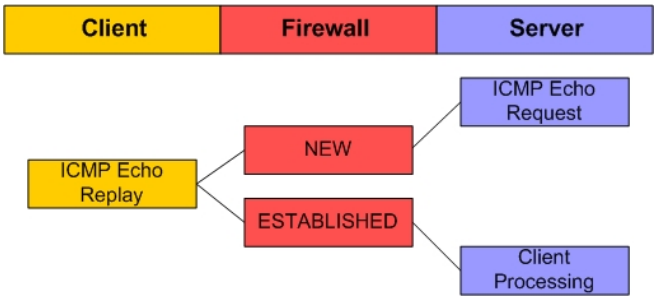
inviato, mentre in fine nel caso di ICMP si considera, ad esempio, il sopraggiungere di una risposta di tipo *echo-reply* (pong) in seguito ad una *echo-request* (ping).

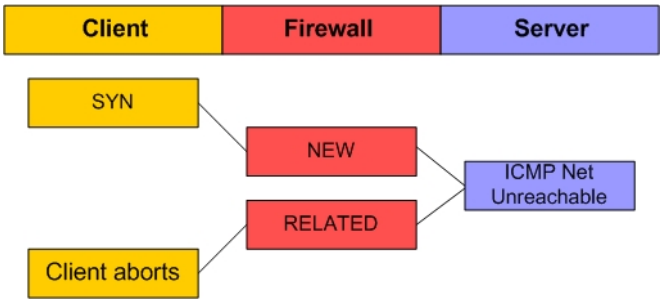




- RELATED Una connessione viene considerata RELATED quando vi è una relazione con un'altra che si trova già nello stato ESTABLISHED. Ad esempio, nel caso del protocollo FTP, una connessione che viene inizializzata dalla porta 20 (ftp-data) è in relazione ad un'altra già attiva sulla porta 21 (ftp-command). Nel caso del protocollo ICMP, una connessione viene considerata RELATED se il pacchetto porta un messaggio di tipo *host-unreachable* o *network-unreachable* riportando quindi un errore di una connessione TCP o UDP già instaurata;

- INVALID Se il pacchetto non può essere identificato diversamente non trovandosi in alcun altro stato viene marcato come INVALID.





2.4 Le catene speciali

Quando avviene il match tra una regola e un pacchetto, l'utente che configura IPTables è tenuto a specificare che sorte fare seguire al pacchetto stesso attraverso un'operazione di jump.

Un pacchetto può essere quindi inoltrato su un'altra catena, accettato o eliminato. Vengono preconfigurate all'avvio le seguenti catene:

- ACCEPT Il pacchetto smette di attraversare la catena e le altre

presenti nella tabella corrente e viene direttamente accettato dal sistema e inoltrato all'applicazione interessata ad esso;

- DROP Il pacchetto viene rifiutato dal sistema;
- LOG Alcune informazioni sul pacchetto vengono memorizzate nei log del firewall attraverso syslog. Il pacchetto continuerà quindi il suo percorso all'interno della catena;
- DNAT Quando un pacchetto viene inoltrato nella catena DNAT, è possibile specificare a quale indirizzo IP inoltrare le connessioni (IPTables sostituirà quindi il campo destinazione del pacchetto con quello indicato dell'utente);
- SNAT Quando un pacchetto viene inviato nella catena SNAT, è possibile specificare quale indirizzo IP utilizzare come sorgente del pacchetto. In questo modo sarà possibile mascherare la rete interna privata servendosi di un indirizzo IP pubblico;
- MASQUERADE Si tratta di un particolare tipo di SNAT nel quale l'indirizzo pubblico da impostare come mittente del pacchetto verrà individuato automaticamente dal sistema.

2.5 Le policy

Una volta che un pacchetto finisce di attraversare una catena senza che sia stata intrapresa nel frattempo alcuna azione, ritorna nella catena di partenza, al punto successivo in cui era stato dirottato. La sorte del pacchetto, una volta terminato l'attraversamento delle catene INPUT, OUTPUT o FORWARD, è condizionato dalla policy della catena stessa impostata dall'utente. A questo punto infatti potrà solamente essere accettato oppure rifiutato.

2.6 Le funzionalità avanzate

IPTables mette a disposizione altri utili strumenti per un filtraggio avanzato. Per citare solo i pricipali:

- Attraverso il *limit match extension* è possibile imporre alcuni generici limiti alle operazioni di jump. Per esempio se ne può trarre vantaggio limitando il logging di una particolare regola nel caso in cui quest'ultima risultasse essere troppo frequente e quindi renderebbe il file di log troppo voluminoso e quindi di difficile interpretazione. In generale però il modulo può essere utilizzato per tutte le tipologie di regole. Sarà quindi anche possibile accettare solo un certo numero di connessioni all'interno di un determinato intervallo di tempo;

- È possibile creare dei filtri che si basano sull'indirizzo MAC (indirizzo fisico della scheda di rete);

- È possibile creare delle regole che tengano conto di connessioni a porte diverse, non necessariamente su di un intervallo numerico contiguo. Inoltre, il fatto di poter specificare degli intervalli di porte e di indirizzi IP differenti, anche nel caso di DNAT, permette di effettuare un semplice ma efficace load balancing tra più server. Ovviamente, per effettuare una ripartizione del carico più efficace, si rende necessario l'utilizzo di strumenti più avanzati di routing inclusi ad esempio nel pacchetto iproute2;

- È possibile creare delle regole sulla base dell'utente o del processo che ha generato il pacchetto. I dati relativi al proprietario vengono identificati attraverso UID, GID e PID;

- È possibile creare dei filtri che si basano sul valore del campo ToS (*Type of Service*) dell'header del pacchetto IP. Questo campo può permettere, tra le altre cose, di gestire il controllo del flusso dati. Infatti potrebbe capitare che un protocollo richieda alla rete che il pacchetto venga consegnato garantendo il minimo ritardo o il minimo costo oppure anche la massima affidabilità. Raramente però queste modalità vengono gestite correttamente dai diversi router.

È possibile rifiutare un pacchetto inoltrandolo sulla catena REJECT al posto della classica DROP. In questo modo sarà possibile specificare la modalità con cui si deve rispondere all'host che ha effettuato la richiesta. Un esempio classico consiste nell'inviare un pacchetto TCP di tipo *tcp-reset* (quindi con il bit RST attivo) qualora si voglia chiudere in modo pulito un tentativo di connessione (in caso contrario, l'host che ha tentato di effettuarla, non ricevendo alcuna risposta, rimarrà in attesa fino al sopraggiungere del timeout) [6].

2.7 Hardening the system

Con "*hardening the system*" si intende la procedura utilizzata per rendere il sistema il più resistente possibile ai vari tentativi di attacco. La procedura di per sé è abbastanza semplice, e si può riassumere in quattro semplici passi:

- Eliminare tutto ciò che è inutile al funzionamento del firewall, ovvero tutto ciò che potrebbe essere utilizzato contro il sistema;

- Rendere difficilmente identificabile il sistema in modo da rendere più difficile per un attaccante capire che strumenti utilizzare per potervici entrare;

- Limitare le capacità del firewall in maniera da rendere la vita il più difficile possibile per un intruso (dischi readonly, mancanza di tool

basilari, strette regole di firewalling anche per la catena in OUTPUT...);

- Mettersi nella posizione di poter rilevare facilmente un'intrusione.

L'idea di base è quella di eliminare tutto ciò che in rete potrebbe essere facilmente attaccato. Occorre quindi seguire le indicazioni contenute nelle prossime sezioni.

2.7.1 Directory /etc/init.d/

In questa directory sono normalmente contenute due categorie di script:

- quelli necessari per effettuare alcune operazioni durante l'accensione e lo spegnimento del computer;
- quelli necessari per far partire o bloccare quei servizi che una volta attivati dovrebbero continuare a funzionare finchè non esplicitamente bloccati (facendo il logout non si fermano – demoni come apache, bind, sendmail, ecc.).

Red Hat 9.0 ha un sistema di boot conforme a quello di System V, in /etc (o /etc/rc.d) esistono altre directory chiamate rc0.d, rc1.d, rc2.d e così via. Queste directory contengono dei link simbolici agli script in "init.d" che dovranno essere eseguiti quando si passa rispettivamente a runlevel 0, 1, 2 e così via.

Ogni runlevel, poi, può essere utilizzato in modo diverso e gli può venire assegnato un particolare significato. Ad esempio, l'operazione di shutdown consiste nel passaggio dal runlevel corrente al runlevel 0 (per cui, tutti gli script in rc0.d verranno utilizzati per bloccare i vari servizi e per eseguire le operazioni di spegnimento), il runlevel 2 è quello standard che viene eseguito quando si accende il computer, ed il 6 viene usato per il reboot.

Per rendere più sicuro il nostro firewall, dovremmo quindi prodigarci per evitare che durante l'accensione vengano avviati dei servizi che potrebbero essere utilizzati dai malintenzionati per penetrare nel sistema.

Il metodo normalmente utilizzato è quello di eliminare i link simbolici (con un semplice `rm`) relativi a servizi che non ci interessano nella directory relativa al runlevel utilizzato per avviare il sistema.

Per avere un elenco dei programmi attivi che offrono servizi in rete, è possibile utilizzare il comando netstat, con

```
netstat -npla |less
```

che indica le porte rimaste aperte ed i relativi processi.

2.7.2 Demoni

In generale comunque, per i servizi che si lasciano attivi, dove possibile occorre:

- Eliminare banner di benvenuto o comunque quelle informazioni che mostrino il nome e la versione dei programmi utilizzati;
- Controllate mailing list di sicurezza, come per esempio *bugtraq*, per verificare che i demoni lasciati attivi non contengano bug pericolosi;
- non lasciate demoni attivi con privilegi di root.

2.7.3 Ripulire il sistema

- Occorre eliminare tutti i tool di compilazione (make, gcc...), gli header file, o tutti quei tool di debugging (gdb, objdump...) che

potrebbero essere usati contro il sistema;

- Occorre cercare tutti quei programmi noti come “suid” root, cioè che possono prendere i privilegi dell’amministratore una volta caricati (`find / -perm +4000`), e togliete loro questa possibilità valutando però caso per caso;

- Occorre configurare propriamente `fstab`, lasciando solo `/var` scrivibile e lasciando tutte le altre partizioni read-only, limitando persino la creazione di dispositivi o l’esecuzione di file suid root.

2.7.4 Rilevare le intrusioni

Per rilevare le eventuali intrusioni è possibile impiegare snort, che è classificato come un IDS (*Intrusion Detection System*). Le informazioni catturate, da esso, vengono trasferite in un DataBase Mysql e tramite l’interfaccia web ACID vengono visualizzati i risultati dei record memorizzati nel DataBase [7].

2.8 Realizzazione di una Rete con firewall IPTables

Il firewall che ho attivato presso l’Area Sistemi Informativi UNICAM ha una configurazione hardware che può essere così riassunta nelle sue componenti principali:

- processore Pentium 2 300 MHz;

- tre schede di rete, due 3Com e una RealTek Ethernet;

- due hard disk, uno da 8 Gb contenete il sistema operativo Linux e l’altro da 2 Gb contente file di log del sistema e dati.

Il sistema operativo installato è Linux, distribuzione Red Hat 9.0, con il kernel versione 2.4.20 opportunamente configurato.

Il sistema Linux conteneva già il pacchetto IPTables necessario per poter configurare un firewall.

Come vedremo in dettaglio nel paragrafo successivo, esiste un solo file di configurazione (`/etc/sysconfig/iptables`) che può essere modificato con un editor testuale; in alternativa è possibile configurare e gestire completamente via web il firewall, con il software Webmin (http://www.webmin.com).

2.8.1 Reti LAN presenti dietro al firewall

Una rete LAN era collegata all'interfaccia `eth1` del computer e aveva come indirizzi privati `192.168.21.0/24`; tramite l'SNAT agli indirizzi privati venivano associati a un range di indirizzi pubblici: `193.20.4.8.132/24-193.204.8.134/24`, come risulta evidente dalla regola nel file di configurazione presente nell'appendice e nella regola successiva:

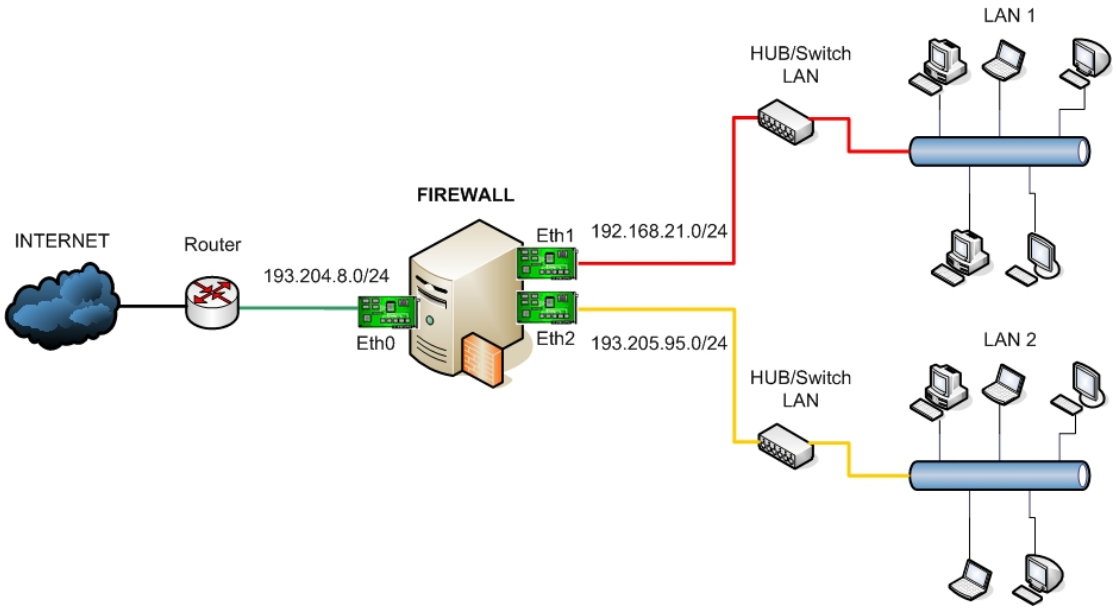
```
-A POSTROUTING -s 192.168.21.0/24 -o eth0 -j SNAT --to 193.204.8.132-
193.204.8.134
```

L'altra che era collegata all'interfaccia `eth2` del computer, invece aveva degli host che avevano degli indirizzi pubblici. La rete era `193.205.95.0/24`; comunque anche il traffico di questa rete era filtrato dal firewall.

Mente internet era collegata all'interfaccia `eth0` e l'indirizzo gli veniva fornito tramite DHCP.

Lo schema di rete con firewall IPTables realizzato nell'laboratorio ASI è il

seguente:



Inoltre ho provato anche ad implementare il masquerading con la seguente regola:

```
-A POSTROUTING  -o eth0 -s 192.168.21.0/24  -j MASQUERADE
```

Tale regola stabilisce che per qualsiasi pacchetto proveniente dalla rete interna `192.168.21.0/24` e che debba uscire dall'interfaccia `eth0`, deve essere effettuato il mascheramento dell'indirizzo mittente (`-j MASQUERADE`).

Comunque per rendere operativo il forwarding è necessario anche invocare un file con il parametro 1, in quanto di default Linux non permette che i pacchetti vengano passati tra le diverse interfacce, ovvero:

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

Quando il valore è `0`, infatti significa proprio che il forward tra le interfacce è disattivo. Tale comando occorre invocarlo prima che vengano caricate le regole di IPTables [8].

Gli Indirizzi IP alle macchine connesse alle due reti private veniva assegnati

tramite un server DHCP. Il file di configurazione (`/etc/dhcpd.conf`) da me configurato è il seguente:

```
ddns-update-style ad-hoc;
option domain-name-servers 193.204.8.13;
option domain-name "unicam.it";
default-lease-time 259200;
max-lease-time 518400;

# Configurazione delle Subnet

# Rete Ufficio
subnet 192.168.21.0 netmask 255.255.255.0 {
      option routers 192.168.21.1;
      range 192.168.21.100 192.168.21.200;
      }
# Rete Ufficio 2
subnet 193.205.95.0 netmask 255.255.255.0 {
      option routers 193.205.95.1;
      range 193.205.95.2 193.205.95.50;
```

Come risulta evidente dal file, il DNS per la risoluzione degli indirizzi è all'indirizzo `193.204.8.13` e il nome di dominio della rete è `unicam.it`.

Ho specificato inoltre il tempo di *lease* quantificato in tre giorni; è il tempo massimo per cui il client può utilizzare l'indirizzo affittato.

Inoltre sono evidenti i range di indirizzi assegnabili ai computer che si connettono alle due reti (`Rete Ufficio` e `Rete Ufficio 2`) con i relativi indirizzi IP dei option routers [9].

2.8.2 Configurazione del firewall

È possibile effettuare la configurazione del firewall in due modi differenti; il primo consiste nello scrivere le regole, presenti nell'appendice, tramite un editor di testo come Vi, l'altro invece consiste nel utilizzare un'interfaccia web come Webmin.

2.8.2.1 Configurazione mediante l'editor Vi

È presente un solo file, `/etc/sysconfig/iptables`, che contiene l'intera configurazione del firewall. Per poter impostare le varie regole, che formano il sistema firewall, ho editato il file citato mediante l'editor Vi.

Le politiche di default per le catene di INPUT, OUTPUT e FORWARD sono impostate ad ACCEPT, se non specificato diversamente. Quindi con le seguenti regole ho chiuso tutto il traffico in INPUT, OUTPUT e FORWARD.

```
:FORWARD DROP [0:0]
:INPUT DROP [0:0]
:OUTPUT DROP [0:0]
```

e poi successivamente ho specificato le regole che mi consentissero di far passare solo il tipo di traffico da me stabilito, come risulta evidente dalle regole sottostanti.

Le opzioni che è possibile stabilire per poter definire le regole sono le seguenti:

- -*N* crea una nuova catena;
- -*A* aggiunge una regola ad una catena;
- -*i* consente, in una regola, di discriminare i pacchetti in base all'interfaccia fisica da cui sono entrati;
- -*o* in base all'interfaccia fisica da cui usciranno (scelta in base alle tabelle di routing da noi impostate);
- -*j* di mandare i pacchetti ad un'altra catena.

Abbiamo visto quindi come una regola può identificare dei pacchetti in base alle schede di rete coinvolte. È però possibile utilizzare molti altri criteri, per esempio:

- il protocollo (-p);

- l'indirizzo ip sorgente (-s);
- l'indirizzo ip destinazione (-d);
- se si tratta di un frammento (-f);
- se non lo è (! -f).

Dove ogni protocollo può aggiungere dei criteri di classificazione. Per esempio, se specifichiamo "-p tcp" per indicare il protocollo tcp, possiamo poi dividere i pacchetti in base anche

- alla porta sorgente (–sport);
- alla porta destinazione (–dport);
- ai flag del tcp (–tcp-flags SYN, ACK, FIN...);
- alle opzioni (–tcp-option).

Qui di seguito sono descritte le regole principale che fanno parte della configurazione da me realizzata.

```
-A OUTPUT -p TCP -o eth0 --dport 53  -j ACCEPT
-A OUTPUT -p UDP -o eth0 --dport 53  -j ACCEPT
-A OUTPUT -p TCP -o eth0 --dport 443 -j ACCEPT
-A OUTPUT -p TCP -o eth0 --dport 80  -j ACCEPT
```

Tali regole stabilisco che il computer che implementa il firewall abbia la possibilità di navigare in internet e quindi sono state abilitate la porta 53 per il DNS sia con il protocollo UDP che TCP, la 443 per l'HTTPS e la 80 per l'HTTP.

```
-A FORWARD -p TCP -i eth0 -o eth2 -d 193.205.95.0/24 --dport 80 -j ACCEP
```

Il significato della sintassi, della regola precedente è il seguente: aggiungi (`-A append`) alla catena di FORWARD, una regola per cui se un pacchetto che utilizza il protocollo TCP (`-p`), proveniente dalla scheda di rete `eth0` (`-i`) ed è destinato ad uscire sulla scheda `eth1` (`-o`), con indirizzo di destinazione della rete `193.205.95.0/24` (`-d`), destinato alla porta `80` (`--dport`), deve essere accettato

(`-j ACCEP`).

```
-A INPUT -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEP
```

Tale regola, possibile solo con IPTables e non con gli altri tipi di firewall, fa in modo che i pacchetti di ritorno, che arrivano al firewall, siano effettivamente associati a una richiesta precedentemente effettuata. Infatti IPTables mantiene lo stato di tutte le richieste interne.

```
-A FORWARD -s 0/0 -p tcp  ! --syn --sport ftp -j ACCEPT

-A FORWARD -s 192.168.21.0/24 -p tcp  --sport 1024: -d 0/0 --dport 1024:
   -j ACCEPT

-A FORWARD -s 0/0 -p tcp ! --syn --sport 1024: -d 192.168.21.0/24
   --dport 1024: -j ACCEP
```

Le regole sovrastanti fanno in modo che la rete privata `192.168.21.0/24` abbia la possibilità di effettuare dei trasferimenti di dati tramite il protocollo ftp (*file trasfer protocol*).

Il file `/etc/sysconfig/iptables` completo di tutte le regole è presente nell'appendice.

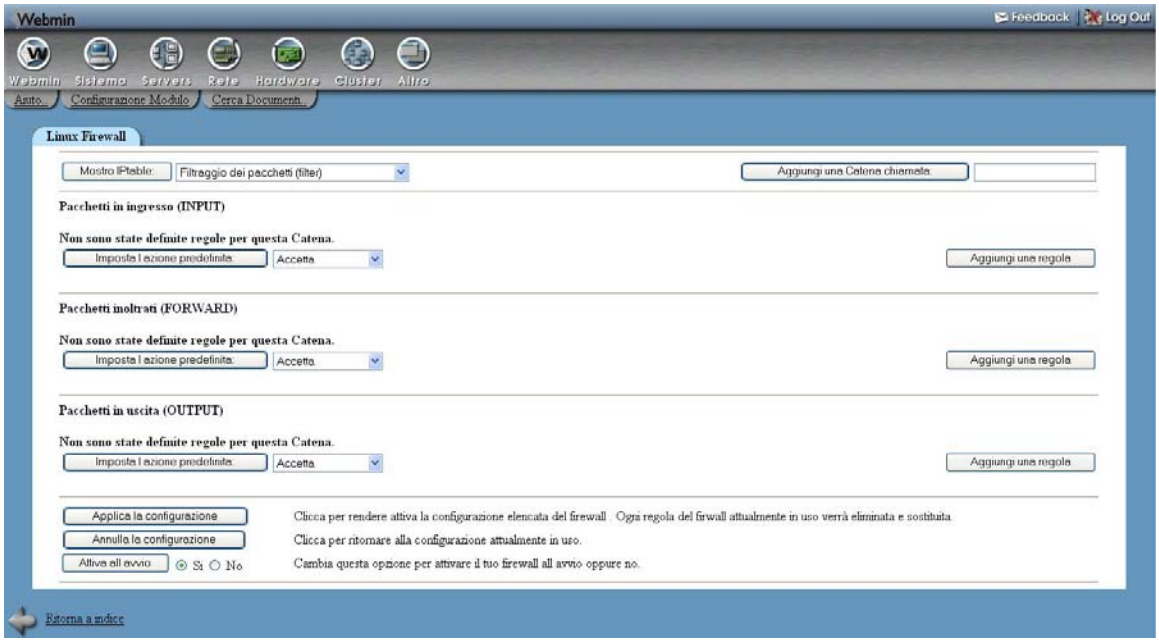
2.8.2.2 Configurazione mediante Webmin

Webmin, è un sistema modulare per la configurazione di tutte le sfaccettature del proprio OS.

Per questo sistema di gestione viene semplicemente utilizzato il Web, per cui l'unica cosa di cui si ha bisogno è un browser, sia esso testuale (come lynx o links) oppure grafico (come il classico Mozilla, Galeon, Konqueror o Netscape). La sua struttura client-server consente agevolmente l'amministrazione da remoto. Il motore di Webmin è un piccolo webserver (sostituibile tra l'altro da Apache, qualora fosse già presente) che utilizza degli script CGI scritti in perl che si occupano della configurazione del sistema. Questo tipo di approccio rende

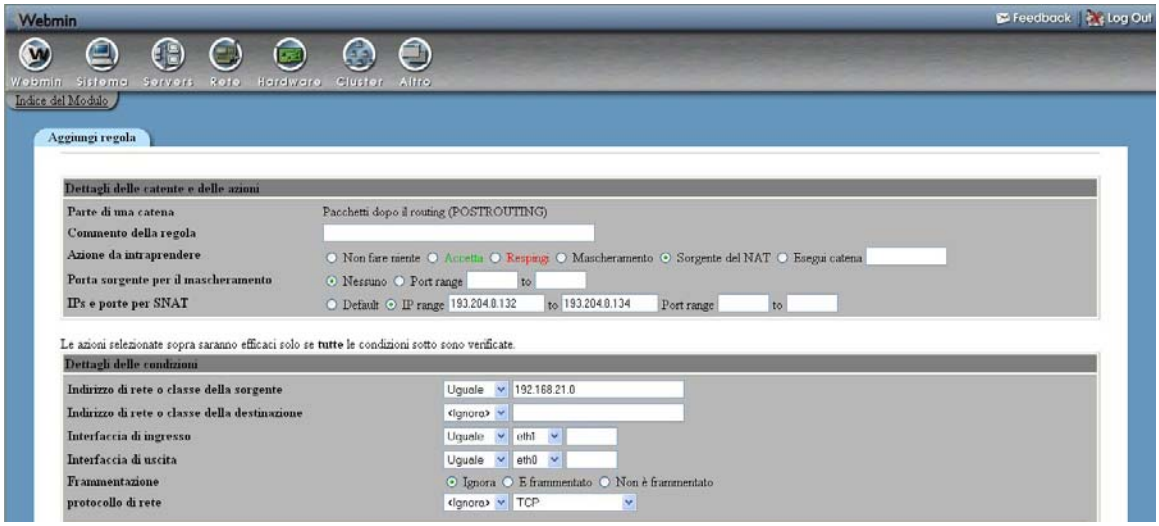
particolarmente modulare questo programma consentendo di “espanderelo” agevolmente utilizzando dei moduli esterni, alcuni sviluppati direttamente dal team di Webmin, altri da terze parti.

Nella figura sottostante viene riportata la schermata iniziale di Webmin in cui è possibile impostare le regole del firewall per le catene di INPUT, FORWARD e OUTPUT.



Nell’immagine successiva invece, utilizzando l’interfaccia di Webmin, viene inserita la regola, sottostante, commentata in precedenza:

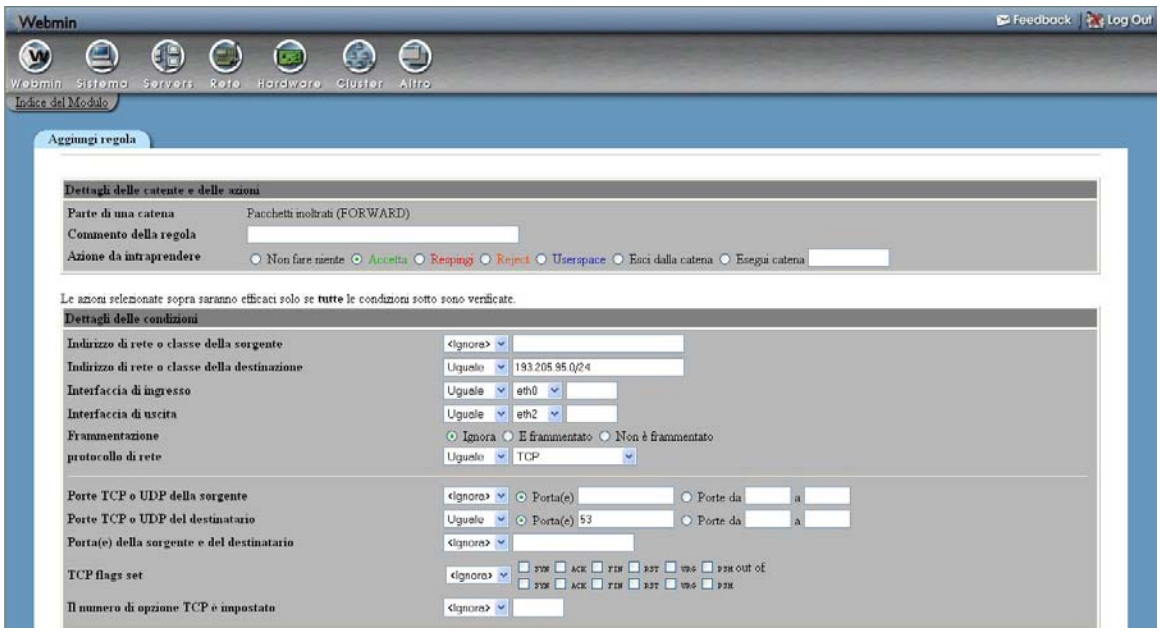
```
-A POSTROUTING -s 192.168.21.0/24 -o eth0 -j SNAT --to 193.204.8.132-
193.204.8.134
```



La seguente regola presente nel file di configurazione ci indica che per la catena di FORWARD i pacchetti TCP provenienti dall'interfaccia `eth0` (Internet), all'interfaccia `eth2` (LAN con indirizzi pubblici) con destinazione la rete `193.205.95.0/24` sulla porta `53` (Domain Name Server) sono accettati.

```
-A FORWARD -p TCP  -i eth0 -o eth2 -d 193.205.95.0/24 --dport 53 –j
ACCEPT
```

L'immagine successiva ci mostra come inserire la regola precedente:



Nell'immagine successiva vengono visualizzare le regole impostate per la catena di INPUT.

Le schermate delle catene di FORWARD e OUTPUT non sono visualizzate. Comunque la visualizzazione è simile all'immagine precedente e le regole impostate sono quelle presenti nell'appendice, che possono essere tutte inserite nel file `/etc/sysconfig/iptables` tramite l'interfaccia Webmin precedente.

2.8.3 *Logging con IPTables*

È possibile tenere traccia dei pacchetti che attraversa il firewall IPTables utilizzando la seguente regola:

```
-A INPUT -m limit --limit 50/minute --limit-burst 3 -j LOG --log-level debug
```

Il significato della sintassi, della regola precedente è il seguente: aggiungi (`-A append`) alla catena di INPUT, `-m limit --limit 50/minute` stabilisce il *maximum average matching rate* impostato a 50 minuti `--limit-burst 3` che riduce il numero di match ad un certo limite per secondo. In definitiva le due

opzioni stabiliscono che il numero massimo di pacchetti, da controllare e quindi inserire nel file di log (`-j LOG`), è di 3 ogni 50 minuti. Tali opzioni sono comodissime col target LOG perché diminuisce la produzione dei log. Mentre l'opzione `--log-level debug` stabilisce la priorità da associare al relativo messaggio di log ossia di debug.

In questo caso il kernel registrerà i pacchetti che passeranno attraverso la catena di INPUT e li associerà ad una chiamata `printk()`. L'output sarà memorizzato nel file di log di linux, simile al seguente formato:

```
Nov 23 14:51:12 localhost kernel: IN=eth0 OUT=
MAC=ff:ff:ff:ff:ff:ff:00:00:e2:76:d4:0d:08:00 SRC=193.204.8.189
DST=193.204.8.255 LEN=229 TOS=0x00 PREC=0x00 TTL=128 ID=31213 PROTO=UDP
SPT=138 DPT=138 LEN=209
```

ovviamente se si imposta un log level più o meno alto è possibile ottenere rispettivamente più o meno informazioni.

Un altro metodo per loggare le informazioni, del traffico scartato, è la seguente:

```
-A INPUT -p icmp -j LOG --log-prefix "ICMP drop:"
-A INPUT -p icmp -j DROP
```

Tali regole indicano al kernel di scrivere nel file di log un messaggio con il prefisso "ICMP drop:" che identificano i pacchetti ICMP che vengono scartati. Ciò permette di individurare i ping effettuati verso il server.

L'output memorizzato nel file di log sarà simile al seguente formato:

```
Mar  9 16:14:30 localhost kernel: ICMP drop:IN=eth0 OUT=
MAC=00:50:ba:5c:f2:1a:00:a0:d1:b6:4e:6f:08:00 SRC=193.204.8.229
DST=193.204.8.69 LEN=60 TOS=0x00 PREC=0x00 TTL=128 ID=20950 PROTO=ICMP
TYPE=8 CODE=0 ID=512 SEQ=1792
```

I messaggi di log precedenti contengono le seguenti informazioni:

- Data e ora del log;
- Nome del server (srv in questo caso);

- kernel: messaggio impostato con il parametero --log-prefix "messaggio";
- IN=nome interfaccia;
- OUT=nome interfaccia;
- MAC=Mac address della scheda;
- SRC=ip sorgente del pacchetto;
- DST=ip di destinazione;
- LEN=lunghezza;
- TOS=valore ToS (prossimamente spiegherò questo tipo di parametrizzazione per la lunghezza e l'ampiezza di banda);
- Proto=protocollo utilizzato (TCP,UDP,ICMP ....)
- SPT: Porta utilizzata dall'indirizzo chiamante;
- DPT: Porta richiesta sulla scheda di rete dall'indirizzo chiamante;

e poi tutti i vari flag.

Solitamente il file di log in cui vengono salvate tali informazioni è `/var/log/messages` tale file è presente sulla root del sistema. Quindi con molta probabilità può capitare che il disco fisso si riempia. Per ovviare a tale problema ho aggiunto un nuovo disco fisso, in cui inserire soltanto il file di log di IPTables.

Per fare questo ho dovuto apportare le seguenti modifiche al file di configurazione di syslog (`/etc/syslog.conf`):

```
*.=debug            /mnt/hd2/log/iptables.log
```

Tali modifiche fanno in modo che i messaggi di log a livello di debug vengano memorizzati nel file: `/mnt/hd2/log/iptables.log`.

2.9 Realizzazione di una Rete con Firewall M0n0wall

Un altro firewall che ho configurato per poter realizzare lo schema di rete descritto precedentemente è M0n0wall (http://m0n0.ch/wall/index.php). Esso è un completo pacchetto firewall open source, che presente le stesse funzionalità di altri pacchetti firewall commerciali e si differenzia da essi per i seguenti motivi:

- È basato su FreeBSD, e non su Linux;
- È ottimizzato per PC non molto potenti, ed è compatibile con un gran numero di periferiche;
- È licenziato su piattaforme come FreeBSD più che GPL 2.

M0n0wall è basato su una versione di FreeBSD, ha un web server (httpd), PHP e altre funzionalità. La configurazione di sistema intera si immagazzina in un file di testo xml per mantenere le informazioni trasparenti.

M0n0wall è probabilmente il primo sistema Unix con il quale è possibile generare la configurazione, mediante un'interfaccia PHP e memorizzare il risultato in un file xml.

Le funzionalità del m0n0wall v1.11 sono:

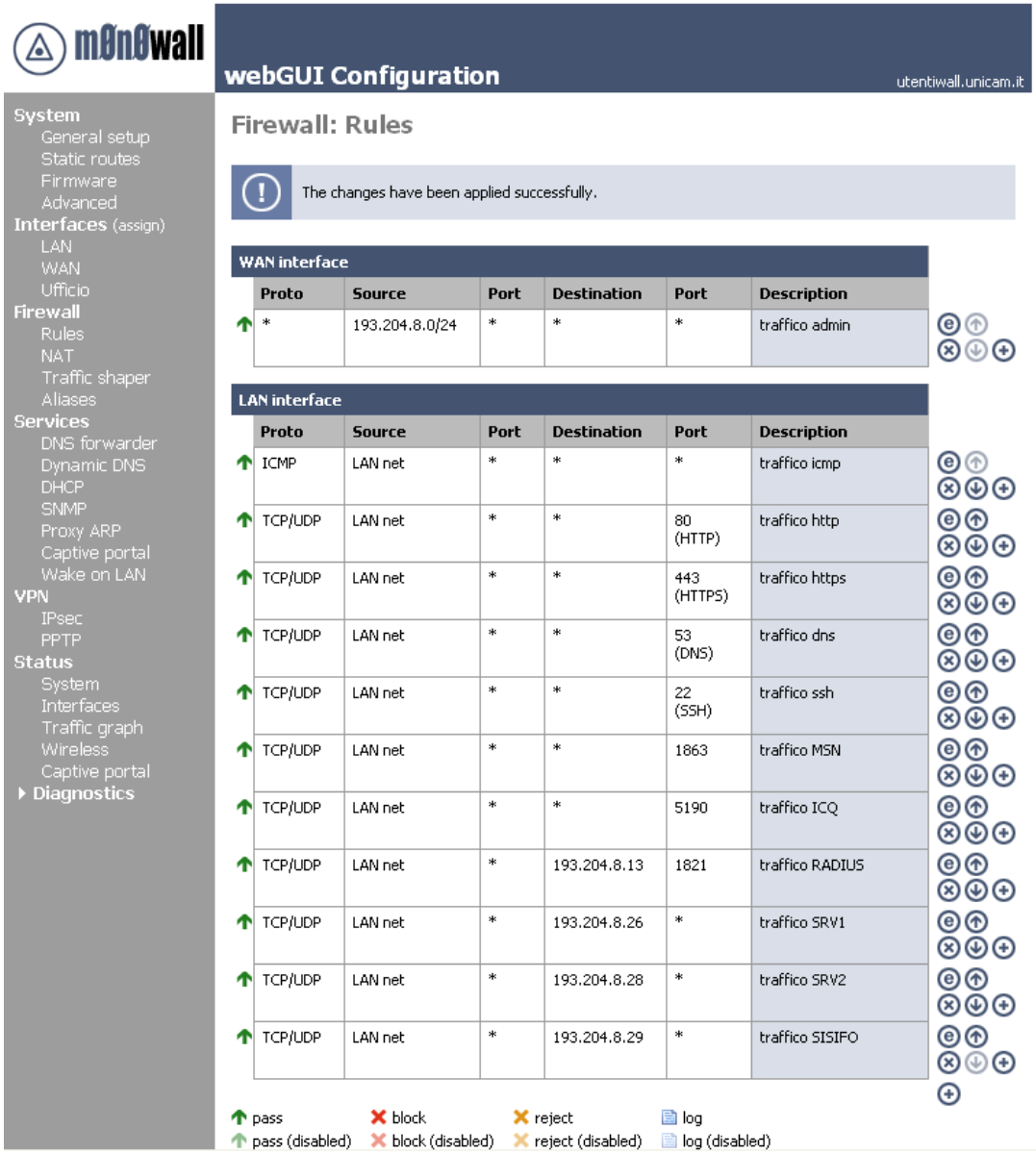
- Filtro pacchetti con regole per permesso/blocco del traffico su tutte le reti;
- NAT & PAT (optional) incluso 1:1;
- Supporto client DHCP, PPPoE, PPTP per le WAN;
- Indirizzi statici;
- Regolatore di traffico;
- Client DNS dinamico;
- Server DHCP, configurabile separatamente per tutte le reti;
- Memorizzazione dei forward DNS con indirizzi di ingresso statici opzionali;
- Supporto per Alias per host e network;
- Supporto per interfaccia Wireless;
- IPSEC VPN endpoint, network to network e clients mobili;
- PPTP VPN endpoint, con supporto autenticazione RADIUS;

- Agent SNMP;
- Logging a un server Syslog remoto;
- Online firmware upgrade;
- Configurazione Backup/Restore.

Il tutto occupa 6 Mb di spazio, che è possibile masterizzare su un cd; il sistema quindi si avvia dal cd e va a leggere la configurazione, da impostare, nel file xml salvato precedentemente in un floppy. Al primo avvio si presenta un'interfaccia per la configurazione delle varie connessioni LAN, WAN e OPT. L'utente può distinguere i vari adattatori in base al loro indirizzo MAC. M0n0wall adotta un concetto di interfacce logiche. Come minimo, devono essere presenti due interfacce di rete, una LAN (Local Network) e una WAN (Wide Area Network) . Tutte le interfacce aggiuntive rilevate vengono viste come OPTx (Optional) dove "x" indica il numero di interfaccia. Queste possono essere rinominate con nomi maggiormente esplicativi mediante l'interfaccia di m0n0wall. Dopo aver configurato m0n0wall, basta effettuare un reboot del sistema e collegare il firewall alla rete privata e a internet. Una volta riavviato, la tastiera e il monitor possono essere scollegati dal firewall.

Successivamente, sarà possibile, mediante un computer presente nella LAN, configurare il firewall comodamente da un browser, mediante l'interfaccia PHP. M0n0wall ci da la possibilità di configurare la tabella di FILTER, quella del NAT e inoltre è possibile anche configurare il Traffic Shaping per poter limitare la banda assegnata a un determinato traffico, su una determinata porta.

Nella seguente immagine è riportato un esempio di configurazione delle regole di filtraggio.



Nell'immagine precedente è possibile vedere le diverse opzioni che compongono la definizione di una regola:

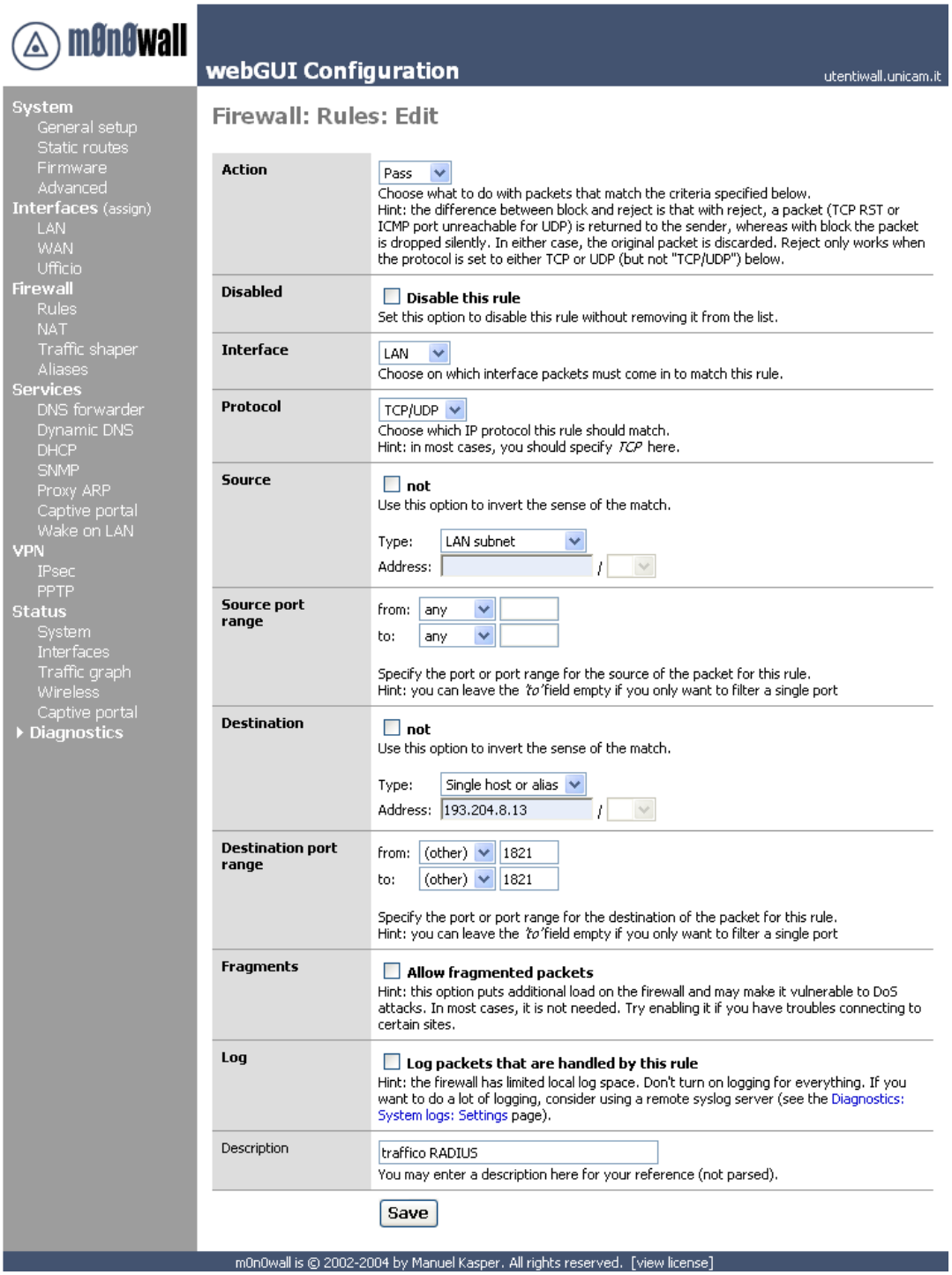
- *Proto* – stabilisce il tipo di protocollo IP, che utilizza il traffico, per quella regola. Le opzioni disponibili sono: (TCP, UDP, ICMP, …., any);
- *Source* – sorgente da cui viene generato il traffico; può essere specificato un singolo host o una rete;

- *Port* – stabilisce il tipo di porta che l'host *Source*, utilizza per inviare il pacchetto;
- *Destination* – destinazione a cui è destinato il pacchetto; può essere specificato un singolo host o una rete;
- *Port* – stabilisce il tipo di porta che l'host *Destination*, utilizza per ricevere il pacchetto;
- *Description* – descrizione della regola specificata;

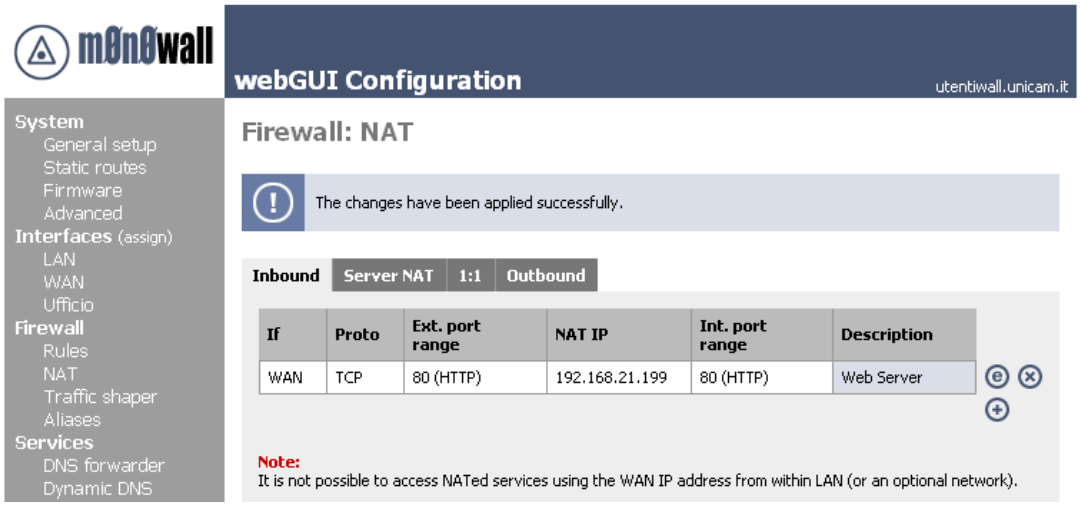
- L'immagine precedente ci fa stabilire invece se il traffico, per quella regola, deve passare, essere bloccato, essere rigettato o semplicemente loggato.

È possibile impostare tali regole sia per il traffico uscente dalla rete LAN (Reti private), sia per il traffico proveniente dalla WAN (Internet).

È possibile impostare le regole che stabiliscono la politica del firewall, mediante la form della schermata successiva, in cui risulta evidente la configurazione della regola per traffico RADIUS:

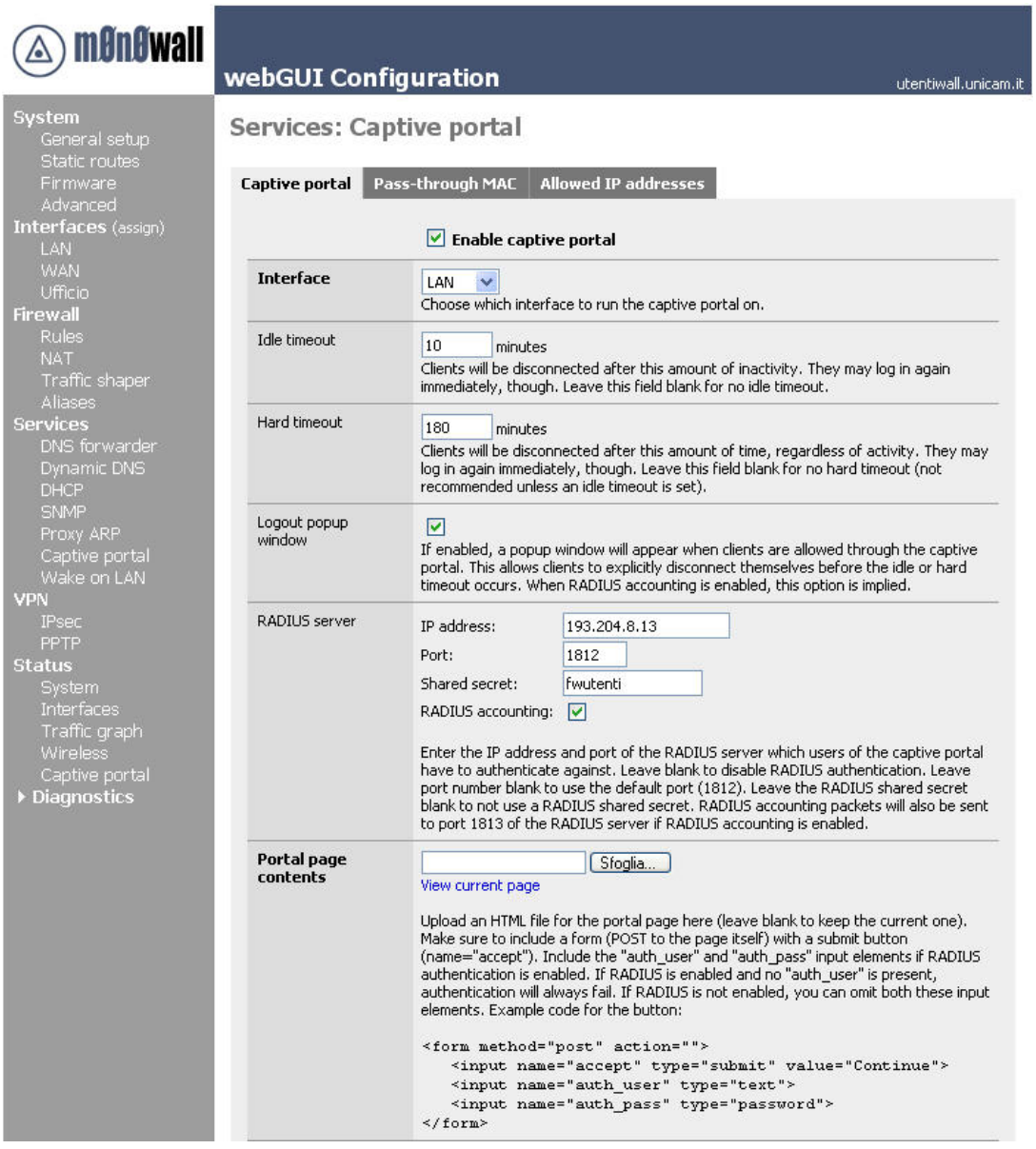


L'immagine successiva invece ci mostra un esempio di configurazione del NAT:



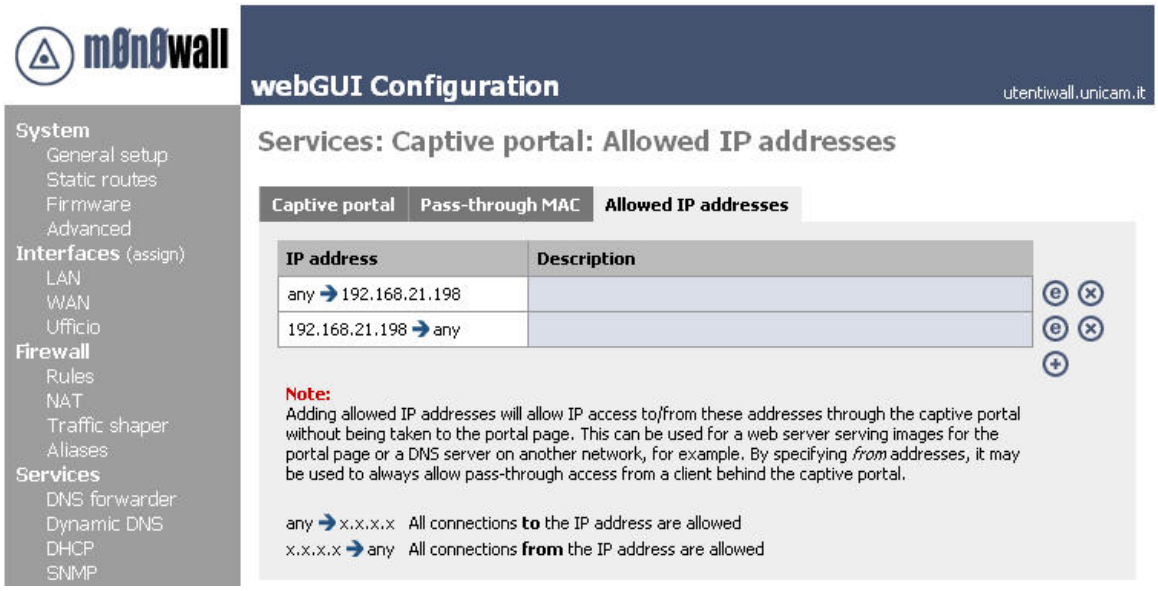
Anche in questo caso c'è un'apposita form per poter impostare la configurazione della schermata precedente, visibile nella schermata successiva:

Inoltre M0n0wall ha la possibilità di interagire con un server Radius, nel nostro caso con ICRadius e fare in modo che tutti gli utenti, che vogliono utilizzare la rete, abbiano un account con cui autenticarsi. È possibile anche impostare l'intervallo entro il quale l'autenticazione risulta valida. Questa funzionalità è visibile nell'immagine successiva.



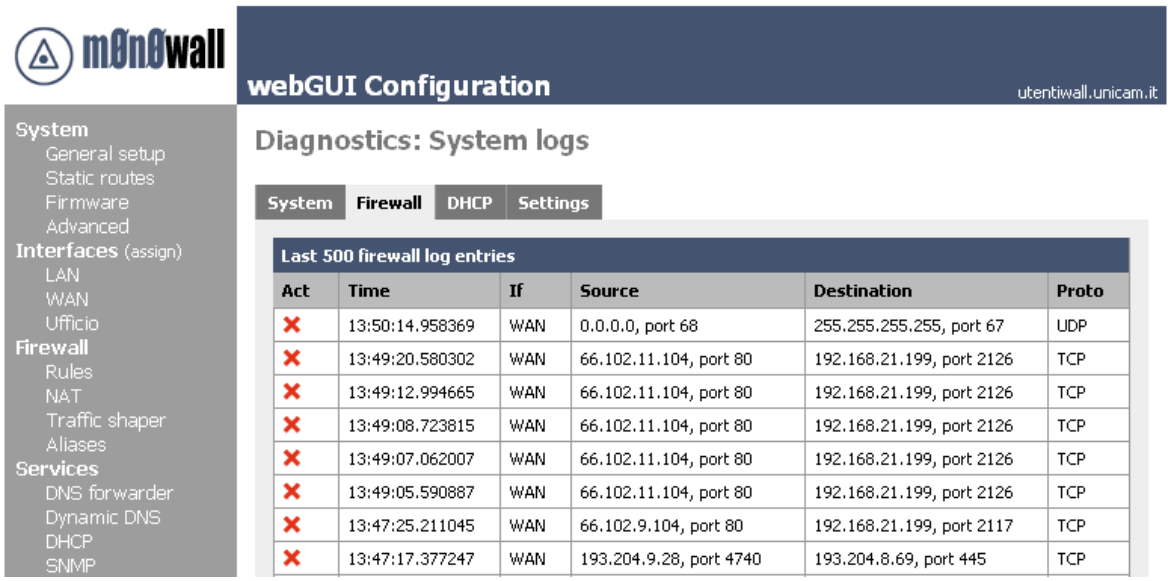
Come risulta evidente dall'immagine sovrastante il server Radius è installato sulla macchina con indirizzo `193.204.8.13` e risponde sulla porta `1812`.

Mentre l'immagine sottostante ci mostra gli indirizzi IP delle macchine che non sono sottoposte all'autenticazione tramite il Server Radius.



Come risulta evidente anche dall'immagine, l'unica macchia che non è sottoposta all'autenticazione, sia per i pacchetti in entrata che in uscita, ha l'indirizzo IP `192.168.21.198`.

Nell'immagine successiva invece viene riportato come m0n0wall salva le informazioni nel proprio file di log, riguardanti il firewall [10].



2.10 Realizzazione di una Rete con Firewall SmoothWall

Il terzo è ultimo firewall che ho configurato tenendo conto del schema di rete precedente è SmoothWall Express 2.0 (http://www.smoothwall.org/).

SmoothWall è una distribuzione firewall open source, basata su GNU/Linux. Include un subset del sistema GNU/Linux, in tal modo non vi è necessità di installare un sistema operativo. SmoothWall si configura tramite una semplice GUI via browser, e non richiede alcuna conoscenza di Linux per essere installato o configurato. È molto utile nel caso in cui si voglia un buon firewall a costo praticamente nullo: gira infatti su hardware datato, e basta poca RAM e pochissimo disco fisso.

È basato su IPTables e supporta: DMZ, NAT, PAT e port forwarding, VPN, connessioni via LAN, linea telefonica, ISDN o ADSL (alcuni modelli). Attualmente è giunta alla release 2.0. Funziona anche con Fastweb residenziale. L'unica mancanza è che non filtra il traffico in uscita, lascia passare tutto da LAN a WAN. D'altra parte la stessa azienda ha in catalogo un prodotto commerciale con molte più feature.

SmoothWall usa un codice di colori molto intuitivo: GREEN è l'attributo della scheda di rete interna, quella cioè collegata allo switch/hub delle vostra rete locale, RED è la scheda esterna, quella collegata al router. Si può avere anche una seconda rete interna, quest'ultima sarà collegata alla eventuale scheda ORANGE.

Dopo la partenza di SmoothWall, è possibile usare un PC della rete locale per la configurazione, tramite un browser. All'interfaccia GREEN è stato dato

l'indirizzo di rete `192.168.21.1`. Il server http di SmoothWall ascolta sulla porta `81`, perciò occorre puntare il browser a: `http://192.168.21.1:81`

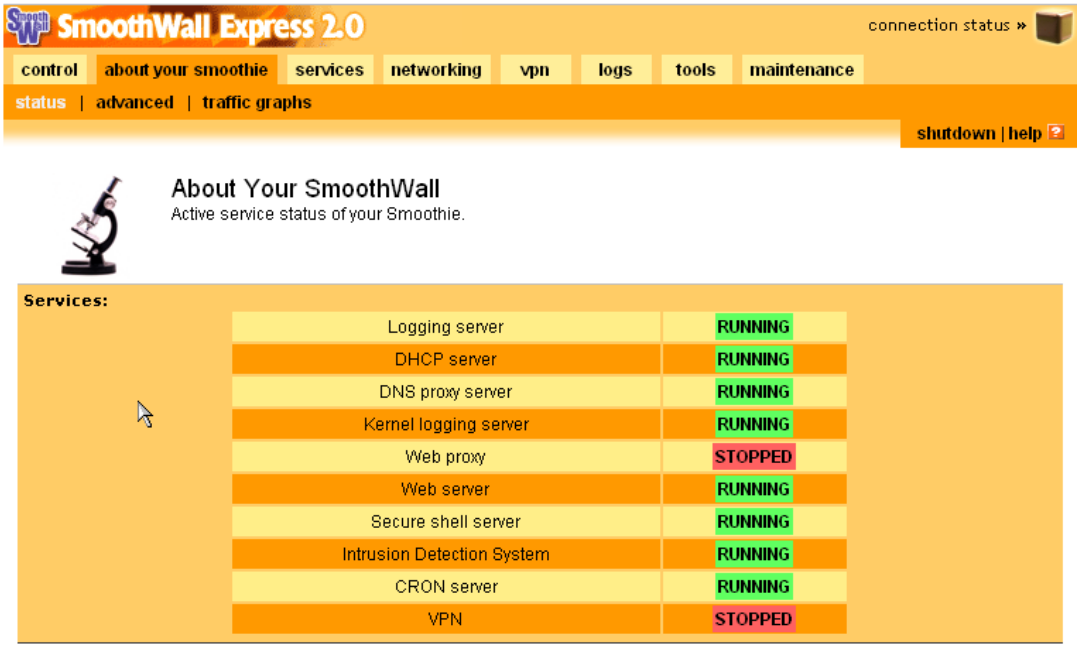
Vediamo nel dettaglio i diversi menu dell'interfaccia di amministrazione:

control

- *home* - pagina principale, da qui potete controllare la versione del prodotto e l'uptime. Se non aggiornate da molto tempo, un messaggio vi ricorderà di farlo;
- *credits* - credits e licenze.

about

- *status* - controllate quali servizi sono attivi sul firewall come è possibile osservare nella seguente immagine:



- *advanced* - informazioni dettagliate sulla memoria in uso e configurazione di partizioni, inode, uptime, interfacce di rete, routing,

moduli e versione del kernel;

- *traffic graph* - grafici basati sulle statistiche di traffico delle interfacce di rete.

services

- *web proxy* - da qui è possibile abilitare e configurare il servizio proxy. Tra le varie opzioni, la possibilità di attivare il proxing trasparente: senza configurare nulla sui client tutto il traffico è automaticamente provato;

- *dhcp* - configurazione e abilitazione del servizio DHCP;

- *dynamic dns* - se non si ha un indirizzo IP fisso, è possibile utilizzare uno dei numerosi servizi che permettono di essere comunque raggiungibili dall'esterno;

- *intrusion detection system* - abilitate SNORT, un IDS che riconosce i tentativi di accesso che sfruttano le falle di sicurezza più note. Serve solo per tenere un log dei tentativi e correre ai ripari nel caso ce ne fosse bisogno. La sicurezza è comunque assicurata dalle regole di forwarding definite più avanti;

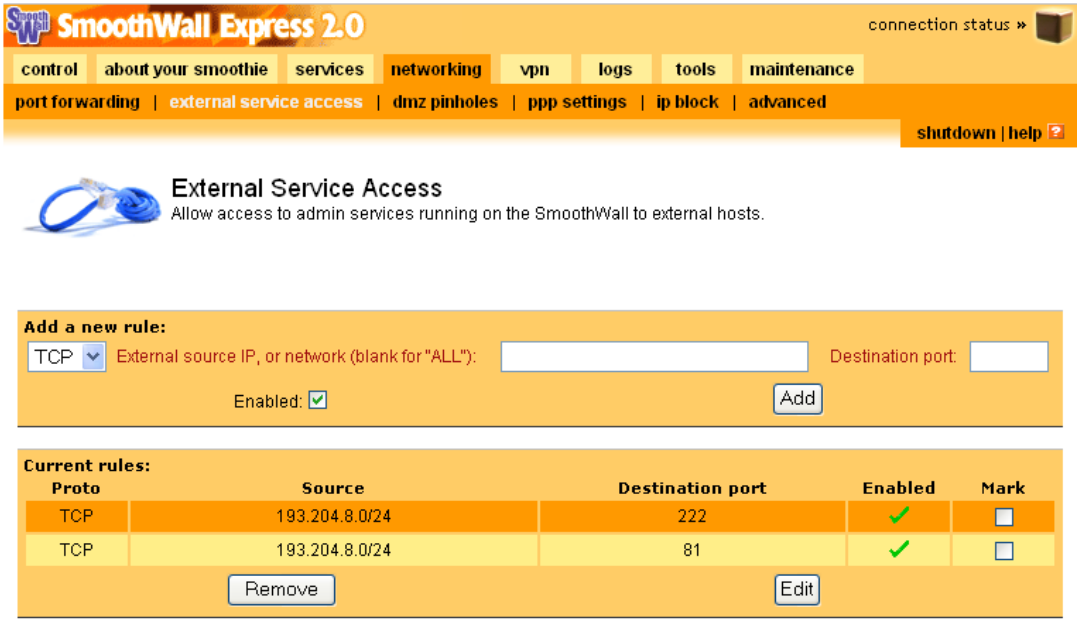
- *remote access* – occorre selezionare entrambe le checkbox, se si vuole gestire il computer sul quale gira SmoothWall via SSH. È molto comodo perché è possibile fare tutta la gestione da remoto senza connettere tastiera e monitor al firewall;

- *time* - cambiate il fuso orario, si imposta l'ora e si configura i server NTP per la sincronizzazione automatica dell'ora da internet. È importante avere data e ora accurati per poter interpretare correttamente i log del sistema.

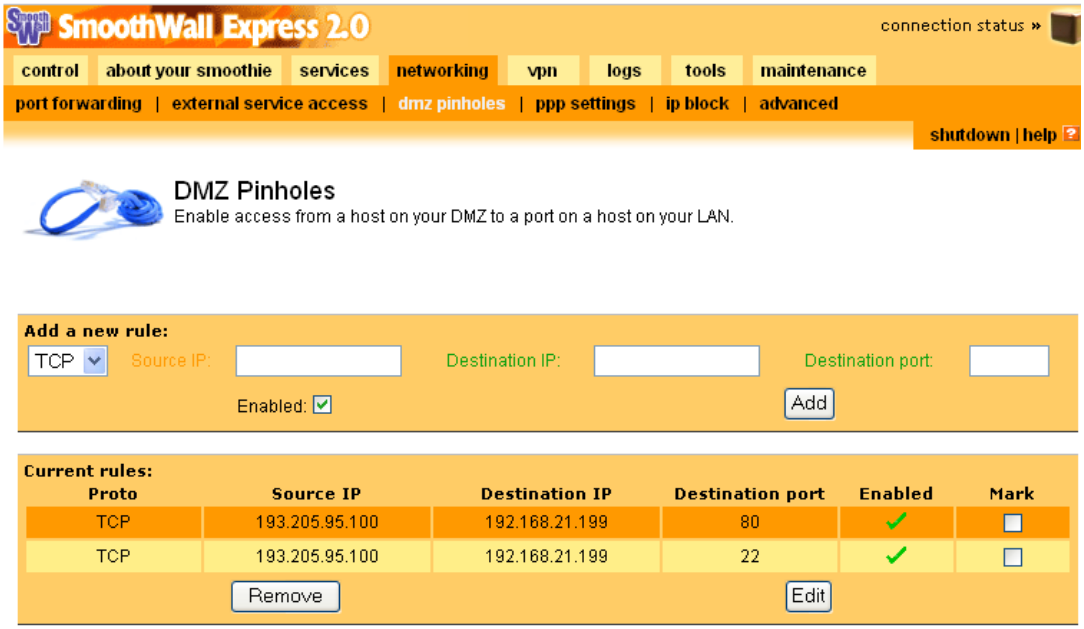
networking

- *port forwarding* - configurazione del port forwarding delle porte TCP, dall'esterno verso le porte TCP dei PC delle LAN;

- *external service access* - per permettere l'accesso dall'esterno a servizi che girano su SmoothWall. Nella figura sottostante è possibile vedere le regole che consentono di controllare il firewall, tramite `http`, dalla rete `193.204.8.0/24` sulla porta `81`, e di accedere in `ssh` sempre per il controllo del firewall tramite la porta `222`.



- *dmz pinholes* - in questa pagina è possibile configurare l'accesso alla LAN (green) dall'altra LAN (orange), solo per determinate porte e host. Come è possibile vede nell'immagine sottostante, in cui è abilitato il traffico dei pacchetti `TCP` sulle porte `22` e `80` dall'host `193.205.95.100` all'host `192.168.21.199`.



- *ppp settings* - serve per configurare l'accesso ad internet via PPP, PPP over ATM o PPP over Ethernet;
- *ip block* – è possibile blacklistare uno o più indirizzi IP;
- *advanced* - configurazione di ICMP, IGMP, UPnP, SYN cookies e traffico multicast, attivando tutto tranne UPnP. Il firewall sarà invisibile dall'esterno e non risponderà alle richieste di ping.

vpn

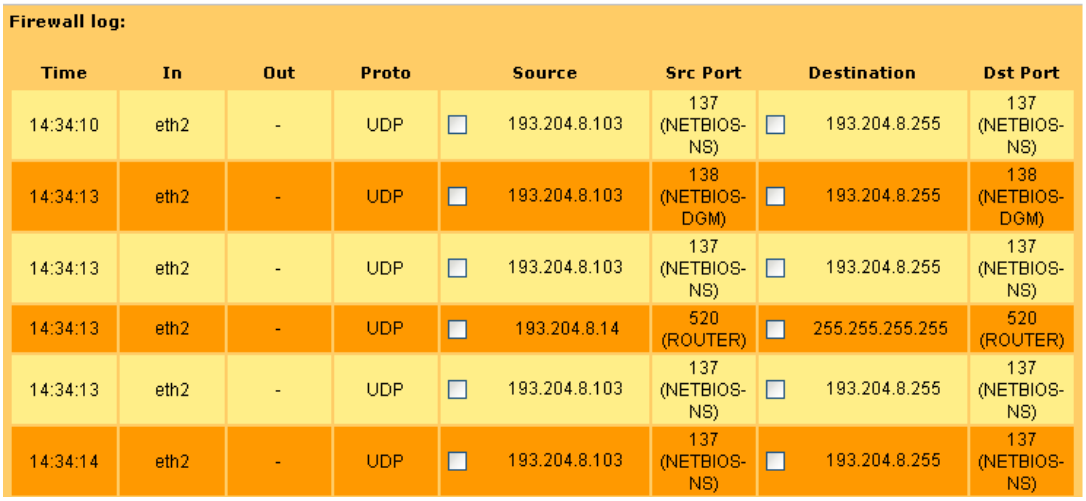
- *control* - SmoothWall può stabilire connessioni VPN con altri SmoothWall, oppure con endpoint FreeS/WAN compatibili;
- *connections* - da qui vengono configurate le connessioni.

logs - questa linguetta contiene tutte le sezioni dedicate ai log del sistema: sono molti e dettagliati.

- *other*
- *web proxy*

firewall

Nella seguente immagine c'è un esempio del log generato dal firewall:

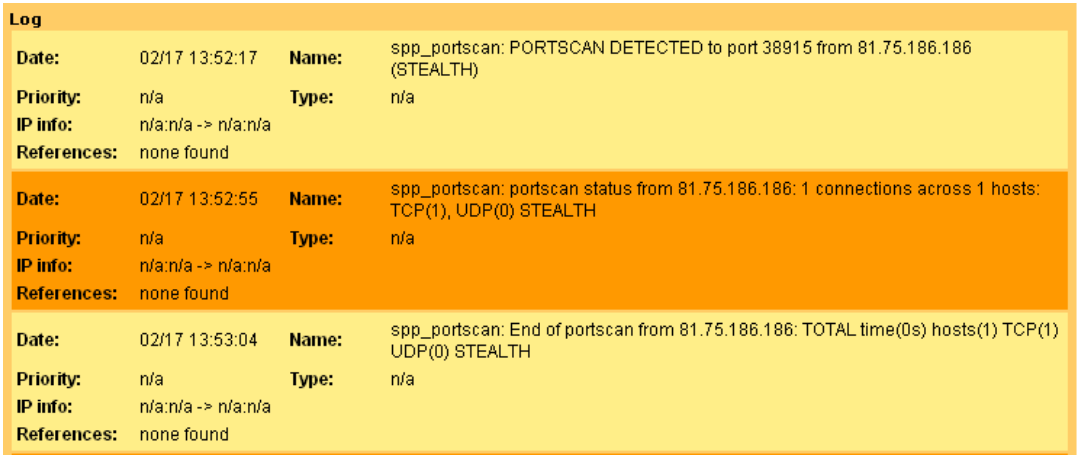


Firewall log:

Time	In	Out	Proto	Source	Src Port	Destination	Dst Port
14:34:10	eth2	-	UDP	193.204.8.103	137 (NETBIOS-NS)	193.204.8.255	137 (NETBIOS-NS)
14:34:13	eth2	-	UDP	193.204.8.103	138 (NETBIOS-DGM)	193.204.8.255	138 (NETBIOS-DGM)
14:34:13	eth2	-	UDP	193.204.8.103	137 (NETBIOS-NS)	193.204.8.255	137 (NETBIOS-NS)
14:34:13	eth2	-	UDP	193.204.8.14	520 (ROUTER)	255.255.255.255	520 (ROUTER)
14:34:13	eth2	-	UDP	193.204.8.103	137 (NETBIOS-NS)	193.204.8.255	137 (NETBIOS-NS)
14:34:14	eth2	-	UDP	193.204.8.103	137 (NETBIOS-NS)	193.204.8.255	137 (NETBIOS-NS)

intrusion detection system

Nella seguente immagine c'è un esempio di log generato dall'IDS Snort:



Log

Date:	02/17 13:52:17	**Name:**	spp_portscan: PORTSCAN DETECTED to port 38915 from 81.75.186.186 (STEALTH)
Priority:	n/a	**Type:**	n/a
IP info:	n/a:n/a -> n/a:n/a		
References:	none found		
Date:	02/17 13:52:55	**Name:**	spp_portscan: portscan status from 81.75.186.186: 1 connections across 1 hosts: TCP(1), UDP(0) STEALTH
Priority:	n/a	**Type:**	n/a
IP info:	n/a:n/a -> n/a:n/a		
References:	none found		
Date:	02/17 13:53:04	**Name:**	spp_portscan: End of portscan from 81.75.186.186: TOTAL time(0s) hosts(1) TCP(1) UDP(0) STEALTH
Priority:	n/a	**Type:**	n/a
IP info:	n/a:n/a -> n/a:n/a		
References:	none found		

tools

- *ip information* - whois su indirizzi IP o domini;
- *ip tools* - ping e traceroute;
- *shell* - se non si vuole usare un client, si può collegare alla console del firewall via SSH.

maintenance

- *updates* - per scaricare ed installare gli aggiornamenti del sistema. Inoltre visualizza lo storico degli aggiornamenti installati;
- *modem* - nel caso si usi un modem per la connessione, qui è possibile trovare la configurazione, sia per PSTN che ISDN;
- *alcatel speedtouch usb adsl firmware upload* - se si ha un modem USB ADSL Alcatel Speedtouch, è possibile aggiornare il firmware dell'apparecchio;
- *passwords* - gestione delle password degli utenti *admin* e *dial*;
- *backup* - per salvare la configurazione su un floppy, per poter ripetere rapidamente l'installazione;
- *shutdown* - per spegnere o riavviare SmoothWall.

Il kernel di SmoothWall viene costantemente aggiornato. Dopo aver configurato le regole di accesso, è possibile controllare la configurazione in dettaglio con il comando "`iptables -L`", da impartire nella console di sistema. È possibile anche modificare le regole di IPTables "a mano" [11].

2.11 Pregi e Difetti dei Firewall Configurati

I tre firewall, open source, che ho avuto modo di configurare hanno tutti dei pregi e dei difetti:

IPTables: è uno dei firewall più flessibili e completi, infatti è possibile specificare con molta accuratezza le regole che gestiscono il traffico dei pacchetti

IP per le catene di INPUT, FORWARD e OUTPUT.

Comunque per poterlo configurare adeguatamente, senza lasciare falle di sicurezza, occorre conosce approfonditamente la sintassi, per la creazione delle regole, oltre a conosce come lavora il protocollo TCP/IP e quindi, quali porte e come le utilizzano i vari servizi.

Inoltre per poter utilizzare tutte le funzionalità messe a disposizione da IPtables occorre installare una distribuzione Unix-Like, nel nostro caso Red-Hat 9.0, su un elaboratore. Ciò comporta, a differenza di altri firewall che non vengono installati su disco, anche dei problemi inerenti la sicurezza del sistema. Infatti se un malintenzionato riuscisse a compromettere la sicurezza del firewall, potrebbe alterare e quindi compromettere il suo funzionamento; tali modifiche quindi verrebbero caricate ad ogni avvio del sistema.

Inoltre è parte integrante di qualsiasi distribuzione Unix-Like quindi è possibile utilizzarlo su tutte le piattaforme che supportano un sistema Unix-Like.

M0n0wall: è un completo firewall open source, che presenta le tipiche funzionalità dei firewall commerciali. È ottimizzato per PC non molto potenti, ed è compatibile con un gran numero di periferiche; inoltre non necessità di un'installazione su disco. Tutto il sistema occupa infatti, non più di 6 Mb ed è possibile caricarlo comodamente da un cd. Questo comporta che se un'eventuale malintenzionato riuscisse a compromettere il firewall, basta far ripartire il computer per riavere il firewall come in principio.

È possibile gestire il suo funzionamento tramite una comoda ed intuitiva interfaccia web. Per poterlo utilizzare e quindi configurare opportunamente, occorre soltanto conoscere come lavora il protocollo TCP/IP e quindi, quali porte e come le utilizzano i vari servizi.

SmoothWall: è una firewall open source, basata su GNU/Linux. Include un

subset del sistema GNU/Linux, quindi non necessita di un'istallazione del sistema operativo. È basato su IPTables ed è configurabile tramite una semplice GUI via browser e non richiede alcuna conoscenza di Linux per essere installato o configurato. È un buon firewall a costo praticamente nullo: gira infatti su hardware datato, e basta poca RAM e pochissimo disco fisso. Per l'installazione necessita di un disco fisso dedicato esclusivamente ad esso e quindi per quanto riguarda la sicurezza se un'eventuale malintenzionato riuscisse a compromettere la sua sicurezza, potrebbe alterare e quindi compromettere il suo funzionamento; tali modifiche quindi verrebbero caricate ad ogni avvio del sistema.

È provvisto di un IDS passivo, ossia provvede soltanto a rilevare eventuali attacchi e a memorizzare il tutto su file di log, facilmente consultabile, tramite interfaccia web.

Comunque tramite l'interfaccia è possibili soltanto creare regole per le catene di INPUT e FORWARD, mentre la catena di OUTPUT accetta qualsiasi traffico. Per poter impostare delle regole per la catena di OUTPUT occorre una shell per poter inserire delle regole con la sintassi di IPTables.

	Hardware Richiesto	**Necessità Disco Fisso**	**Open Source**	**Basato su:**	**Interfaccia di Programmazione**	**Catene Principali, Modificabili**
IPTables	Computer non troppo vecchio	Si, partizionato	Si	Sistema Unix Like	Riga di comando, o interfaccia web (Webmin)	INPUT FORWARD OUTPUT
M0n0Wall	Computer anche vecchi senza disco fisso	No	Si	FreeBSD	Interfaccia web PHP proprietaria	INPUT FORWARD OUTPUT
SmoothWall	Computer anche vecchi con disco fisso	No, dedicato	Si	Linux IPTables	Interfaccia web proprietaria	INPUT FORWARD

CAPITOLO 3

INTRUSION DETECTION SYSTEMS

3.0 Introduzione agli IDS

3.0.1 La sicurezza informatica

La sicurezza informatica è argomento assai vasto e complesso che comprende svariati problemi.

Un attacco ad un sistema informatico può avvenire in diversi modi ed anche le azioni per la difesa dei sistemi informatici dagli attacchi assumono svariate forme. Viene riportata una piccola classificazione dei tipi di attacco ed intrusione ai sistemi informatici e le varie tecniche utilizzate per la salvaguardia degli stessi. Questa breve nota servirà ad introdurre gli IDS ed a capire il loro fondamentale compito.

3.0.2 Tipi di intrusione

Secondo la classificazione fatta da Anderson [12] agli albori del problema sicurezza e poi ripresa da Axelsson nelle sue ricerche sugli IDS [13] [14] si possono identificare quattro classi di possibili attacchi, a seconda della loro provenienza (vengono lasciati i nomi in inglese per una maggiore chiarezza):

- *External Penetrator*: colui che accede fisicamente al computer a cui non dovrebbe avere accesso;
- *Masquerader*: colui che riesce ad accedere ad un sistema autentificandosi come utente autorizzato;

- *Misfeasor*: colui che, pur avendo diritto di accedere a risorse protette, abusa dei suoi privilegi attaccando la sicurezza del sistema;
- *Clandestine user*: colui che riesce ad accedere al sistema autenticandosi come amministratore del sistema.

Un modo alternativo per classificare un attacco informatico è in base a quale requisito della sicurezza del sistema viene a mancare come effetto dell'attacco. I principali sono tre:

- *Confidentiality*: si vuole impedire ad estranei di accedere a propri dati privati, cioè si vuole tutelare la segretezza delle informazioni;
- *Integrity*: si vuole impedire ad estranei di modificare dati privati, cioè si vuole salvaguardare l'integrità delle informazioni;
- *Availability*: si vuole impedire a chiunque la possibilità di rendere inaccessibile il sistema, cioè si vuole tutelare l'efficenza del sistema.

Le due classificazioni non sono in contrasto ma si vedrà che nel caso degli IDS varrà utilizzata per lo più la prima, in particolare si tratterà il caso di attacco da parte di un Masquerader.

3.0.3 Intrusion Detection Systems

Con *Intrusion Detection Systems* o più brevemente IDS si denominano tutti quei sistemi per il rilevamento di intrusioni all'interno di un computer o di una rete informatica.

3.0.3.1 Caratteristiche degli IDS

Gli IDS devono funzionare come un allarme: controllare lo stato del sistema e, nel caso si riscontrino situazioni considerate pericolose, avvertire l'amministratore della rete informatica del problema riscontrato. Contrariamente a quanto avviene per gli allarmi fisici utilizzati nelle abitazioni, gli IDS devono distinguere accessi ed operazioni consentiti, da altri probabilmente fraudolenti. È questo è il punto caldo contro cui si devono scontrare i costruttori di IDS, perchè spesso gli attacchi avvengono da parte di Masquerader e quindi sono mascherati da azioni permesse. Ci si pone quindi il problema di come agire. Ci sono due tecniche base per "smascherare" l'azione non permessa. Rimane però il problema di trovare delle indicazioni generali che tutti gli IDS devono seguire. Un IDS perfetto sarebbe quello che riesce ad individuare tutte e sole le intrusioni, ma purtroppo l'IDS perfetto non potrà mai essere costruito [15].

Questo perché sarà sempre presente almeno uno di questi due casi:

- *Falsi positivi*: l'IDS rileva qualcosa di anormale ma non è avvenuta alcuna intrusione;
- *Falsi negativi*: l'IDS non rileva un'intrusione avvenuta.

Ovviamente la situazione più grave è la seconda perchè può compromettere gravemente la sicurezza del sistema; ma anche un'eccessiva presenza di falsi positivi potrebbe portare ad un'impossibilità di utilizzo del computer per un eccesso di avvisi di intrusione infondati.

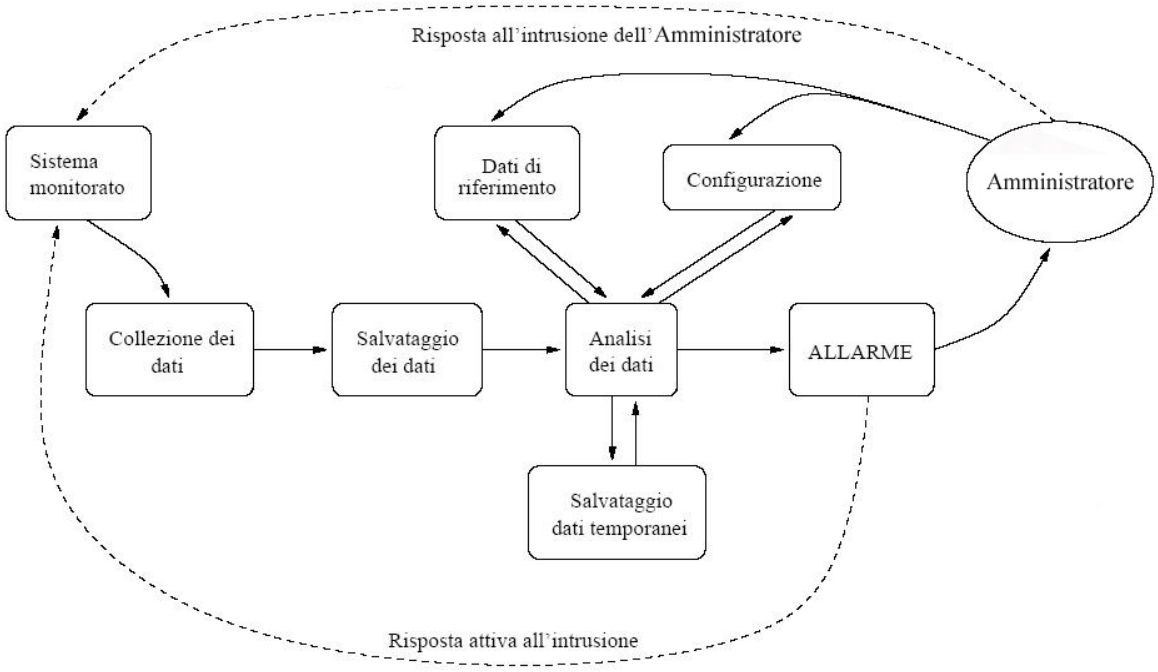
L'obiettivo di ogni IDS è minimizzare falsi positivi e negativi.

Purtroppo spesso una diminuzione di falsi positivi vuol dire meno controllo e quindi più falsi negativi e viceversa.

Un altro problema correlato allo sviluppo degli IDS riguarda la gestione delle risorse; un buon IDS lavora anche mentre il computer è in uso da altri utenti e quindi non può impedire o limitare gravemente l'uso di risorse condivise.

3.0.3.2 Struttura e classificazione generale di un'IDS

Gli IDS sono molto diversi tra loro ma si riesce comunque ad evidenziare una struttura comune a molti di essi. La seguente figura rappresentata l'architettura base di un IDS, è importante notare come possano esistere IDS senza alcuni elementi della figura e sopratutto ogni IDS ha proprie particolarità strutturali.

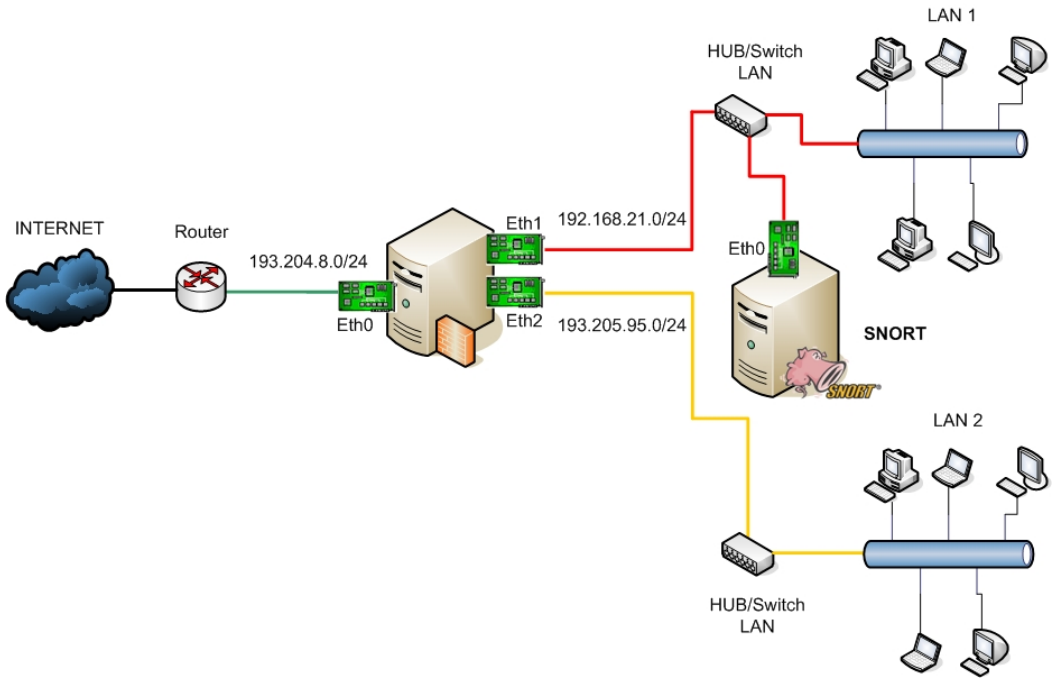


L'amministratore di sistema, che risponde agli allarmi generati dall'IDS, agisce sui file di configurazione e di policy dello stesso, o direttamente sul sistema in caso di bisogno. L'IDS è composto dai moduli "collezione dei dati", "salvataggio dei dati" e "analisi dei dati" e si appoggia a uno storage temporaneo e opera seguendo le istruzioni che l'amministratore ha riportato nei file di configurazione.

3.1 Snort

Snort può essere impiegato su praticamente qualsiasi nodo di rete, con minime interferenze alle operazioni normali. È un sistema multipiattaforma (dovrebbe compilare e funzionare correttamente su tutti i sistemi compatibili con le libpcap ed inoltre è stato effettuato un porting in ambiente Win32) ed è facilmente configurabile, ciò permette di implementare soluzioni di sicurezza specifiche in breve tempo.

L'immagine successiva mostra in quale posizione della rete è stato collocato l'IDS Snort:



3.1.1 *Caratteristiche generali*

Snort è un Network based IDS realtime, *lightweight*, con risposta passiva alle intrusioni e con una ottima accuratezza nel rilevamento delle intrusioni che sfruttano vulnerabilità note.

Snort ha caratteristiche realtime, in quanto è in perenne ascolto sul segmento di rete a cui è stato adibito ed è in grado di contattare l'amministratore di sistema attraverso molteplici meccanismi come syslog e messaggi "Winpopup".

La leggerezza di questo software deriva principalmente da due caratteristiche, la ridottissima dimensione dello stesso (i sorgenti di Snort hanno una dimensione inferiore ai 600KB), e dall'efficienza nella scansione dei pacchetti che lo rende utilizzabile anche su hardware non molto performante.

Snort non è in grado di bloccare i pacchetti considerati maliziosi, si limita a informare l'amministratore con i meccanismi sopra elencati. Questo non impedisce certo di far reagire il sistema in base alle e-mail di notifica ricevute (ad esempio istruendo un firewall a monte di scartare i pacchetti provenienti dall'ip che ha fatto allertare Snort).

L'IDS in questione adotta un meccanismo di *Policy-Detection* attraverso regole che possono essere molto specifiche e ispezionare l'intero campo dati del pacchetto di rete. Questo consente la scrittura di regole che identificano univocamente pacchetti generati per sfruttare attacchi note come i *buffer overflow* [16] o i *cgi-probe*.

Questo software è considerato un IDS di "basso livello" in quanto non consente di effettuare controlli su pacchetti frammentati o direttamete su campi di protocolli di livello applicazione, ma questa sua semplicità si è fino ad ora rivelata una caratteristica che lo ha reso celebre. La semplicità di scrittura delle regole e il basso livello a cui opera lo hanno reso uno strumento particolarmente duttile [17], anche come strumento di analisi di attacchi sconosciuti.

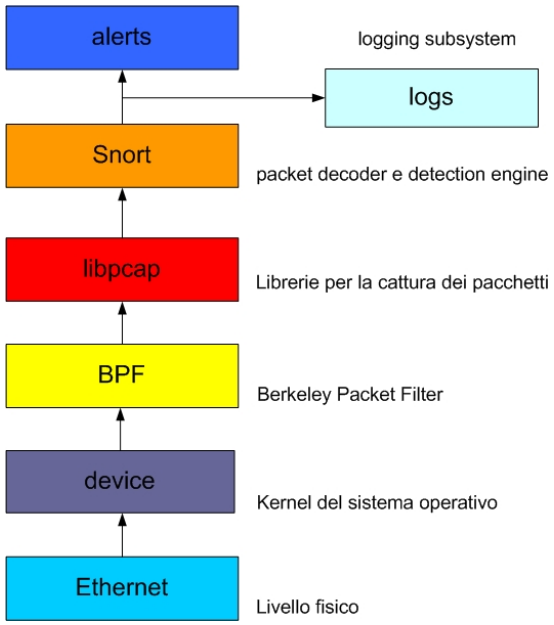
La semplicità del tool e la sua struttura modulare hanno fatto in modo che non dovesse mai essere modificato per poter identificare nuove tipologie di attacco.

Un'altra caratteristica importante di Snort è la sua portabilità, che deriva direttamente dalle librerie di packet sniffing [18] su cui è basato. Attualmente è stato portato su Linux (tutte le architetture supportate), la famiglia *BSD, Windows (sia della serie 9x che NT), Solaris (su Sparc e x86), MacOS X, IRIX e HP-UX.

Ultima caratteristica vincente di questo IDS è la licenza free che ha fatto sì che un nutrito gruppo di sviluppatori abbia contribuito al suo perfezionamento, e ne ha facilitato la diffusione attirando cospicui investimenti. Una prova della bontà di questo software e del suo utilizzo in ambiti altamente professionali è la sua adozione da parte del Ministero della Difesa degli Stati Uniti d'America.

3.1.2 Struttura

La struttura di questo IDS può essere analizzata considerando tre blocchi: il packet sniffer (che è completamente implementato dalle libpcap [18]), il parser per le regole e il motore di analisi dei pacchetti. Nell'immagine successiva viene mostrata la sua struttura semplificata.



3.1.2.1 Algoritmi di pattern matching

Snort è in grado di analizzare il body dei pacchetti in cerca di *signature* note, nel traffico di rete. Questo è un lavoro molto oneroso in termini di cicli di cpu e un buon algoritmo di *pattern-matching* è necessario per avere un tool di buon livello. Snort ha avuto una evoluzione graduale dallo stadio di prototipo, fino all'attuale stato di software robusto e performante.

Come descritto da Neil Desai [19], si possono distinguere tre fasi nello sviluppo della componente di pattern-matching di Snort. Inizialmente veniva utilizzato un semplice *string-compare*, che è la soluzione più semplice e facile da implementare, ma anche la più inefficiente. Il primo passo nello sviluppo del pattern-matcher fu quello di implementare l'algoritmo di Boyer-Moore [20] per la ricerca veloce di sottostringhe. In seguito l'algoritmo divenne un ibrido chiamato Aho-Corasick_Boyer-Moore, che sfruttava alberi ordinati secondo un criterio lessicografico per non ripetere ricerche di stringhe inutili.

Il secondo passo fu di implementare una struttura dati per le regole da applicare, basata su una matrice bidimensionale implementata attraverso liste. Questo permise di organizzare le regole in modo da doverne sempre applicare il minor numero possibile, scartando tutte le regole rese false da ogni controllo effettuato. Questo permise di incrementare le performance di un fattore 100 [19] su sistemi configurati con un grande numero di regole.

Il terzo stadio è consistito nella transizione dalla matrice bidimensionale a una tridimensionale, sempre implementata attraverso liste, che consente l'utilizzo di plugin, ma non incrementa le performance.

3.1.2.2 Le regole

Le rules sono sicuramente la parte più importante e delicata della configurazione di qualsiasi IDS, e Snort non fa eccezione. Anche in questo ambito

vale la solita regola che il miglior prodotto di security del mondo, se mal configurato, non serve al suo scopo. È perciò fondamentale curare particolarmente la formulazione delle regole, per evitare i cosiddetti “falsi positivi” (falsi allarmi derivati dall’eccessiva strettezza delle rules) ed i mancati allarmi (o falsi negativi, derivati al contrario dalla troppa permissività).

È quindi chiaro l’importanza di bilanciare le regole in modo da evitare per quanto possibile queste situazioni estreme.

Ogni rete dovrebbe avere delle regole scritte ad hoc, dipendenti da molti fattori, quali il tipo di traffico, il numero e la tipologia degli utenti, i servizi offerti ecc. Un amministratore trova davanti a sé due strade principali: scrivere le regole personalmente da zero oppure scaricare dei ruleset già pronti e modificarli adattandoli alle proprie esigenze.

Le regole di Snort sono abbastanza semplici da scrivere, ma abbastanza potenti da essere in grado di individuare una grandissima varietà di traffico ostile o anche solo semplicemente sospetto (anomalie e attacchi veri e propri).

Vi sono tre azioni di base che possono essere eseguite nel momento in cui un pacchetto *matcha* una delle regole: pass, log e alert. Pass ignora semplicemente il pacchetto, log scrive il pacchetto utilizzando la routine di jogging selezionata all’avvio di Snort e alert, infine, genera un allarme e notifica l’amministratore nel modo da lui scelto in precedenza.

Le regole basilari contengono solamente le informazioni riguardanti protocollo, direzione e porta interessata, oltre ovviamente all’azione da compiere. Ad esempio:

```
log   tcp   any   any   ->
                    10.10.10.0/24  79
```

Questa regola registrerà tutto il traffico entrante verso la porta `79` (servizio

finger), su tutti gli host della rete rappresentata dalla classe C `10.10.10.x`.

Se non volessimo limitarci a loggare un certo tipo di traffico, ma volessimo anche generare degli alert, dovremmo utilizzare una sintassi più estesa, per specificare il contenuto e la tipologia dei pacchetti per cui far scattare l'allarme.

Prendiamo l'esempio del `phf.cgi`.

```
alert   tcp   any   any   ->
                        10.10.10.0/24  80
               (content:  "/cgi-bin/phf";
                 msg:  "Probe del PHF!";)
```

Grazie a questa regola, Snort ci avvertirà di un possibile attacco tutte le volte che sulla rete da lui monitorizzata passerà un pacchetto diretto ad una macchina nella nostra classe C, contenete una richiesta via web (porta 80) del CGI phf.

È ovviamente importante notare che se, ad esempio, i server utilizzano lo ScriptAlias "cgi" al posto di "cgi-bin" la regola di cui sopra va modificata con la corretta indicazione del path. Proseguiamo nella overwiew, illustrando la possibilità di indicare un range di porte al posto di una porta singola:

```
alert   tcp   any   any   ->
                      10.10.10.0/24  6000:6010
                    (content:  "/cgi-bin/phf";
                      msg:  "Probe del PHF!";)
```

Sia le porte che gli indirizzi IP possono essere modificate tramite l'operatore "!" di negazione, utile ad esempio per individuare tutto il traffico X proveniente dall'esterno della nostra LAN:

```
alert   tcp   !10.10.10.0/24  any   ->
                      10.10.10.0/24  6000:6010
                    (content:  "/cgi-bin/phf";
                      msg:  "Traffico X-Window";)
```

Ovviamente possiamo andare molto oltre, individuando il traffico in formato

binario (utile per il detect degli exploits remoti), grazie agli operatori "||":

```
alert   tcp  any  any   ->
                    10.10.10.0/24  111
                  (content:  "|00 01 86 a5|";
                    msg:  "Accesso al mountd";)
```

Possiamo inoltre filtrare i pacchetti secondo altri canoni e non solo basandoci sull'analisi del contenuto:

```
alert   tcp  any  any   ->   any  any
                   (minfrag: 256; msg:  "Piccoli
                    frammenti individuati, possibile attività ostile";)
```

Successivamente fornisco una tavola delle possibilità principali offerte dal sistema di rules di Snort. Per quanto riguarda l'header delle regole le opzioni a nostra disposizione sono:

selezione delle azioni da compiere (pass, log, alert);

- selezione del protocollo in oggetto (tcp, udp, icmp);
- indicazione degli indirizzi IP e delle eventuali porte;
- selezione dell'operatore direzionale singolo "->" o doppio "<>".

Nel Campo opzioni, invece, troviamo tra le altre cose:

- *msg* (messaggio di alert da inviare in caso di match);
- *logto* (utilizzo di uno speciale file log di output);
- *mingrag*, che come abbiamo detto permette di controllare la frammentazione;
- *ttl*, *id*, *dsize* (controllo approfondito del pacchetto IP);
- *content* (analisi del contenuto, in forma ascii o in binario);

- *offset* e *depth* (per definire quale porzione del pacchetto analizzare);
- *flags* (varie flags TCP, tra cui FIN, SYN, RST, PSH, ACK, URG);
- *icode* (discrimina in base al codice ICMP).

Comunque per scrivere delle buone regole dobbiamo infatti tenere conto dei casi in cui il contenuto dei pacchetti non è realmente necessaria. Quando una sessione passa in status ESTABLISHED, infatti, i flags PSH e ACK sono settati nell'header del pacchetto contenente i dati: è quindi generalmente inutile analizzare il contenuto di pacchetti che non hanno questo flag. Conviene perciò utilizzare una regola del tipo [21]:

```
alert   tcp   any   any   ->
                        10.10.10.0/24  80
        (content:  "/cgi-bin/phf"; flags:
             PA; msg:  "Probe del PHF!";)
```

3.1.3 Prova di utilizzo

Per prima cosa ho avviato Snort sulla macchina con IP `192.168.21.198` con il seguente comando impartito da shell:

```
/usr/sbin/snort –c /etc/snort/snort.conf –l /var/log/snort
–h 192.168.21.0/24 –D
```

Tale comando fa si che Snort vada a loggare su un file, piuttosto che su un database mysql [22].

Tale comando lo avvia tenendo conto delle impostazioni presenti nel file di configurazione `/etc/snort/snort.conf` e vada a salvare il file di log nella directory `/var/log/snort` prendendo in considerazione la rete `192.168.21.0/24`. Mentre l'opzione `-D` fa in modo che Snort agisca come demone senza occupare una

shell.

Per studiare Snort in azione sono stati simulati alcuni semplici attacchi dalla macchina con IP `192.168.21.193` alla macchina con IP `192.168.21.198`.

Il primo esempio mostra la reazione di Snort a una richiesta maliziosa al server web della macchina con IP `192.168.21.198`, fatta dalla macchina con IP `192.168.21.193`.

```
root@192.168.21.193# telnet 192.168.21.198 80
Trying 192.168.21.198...
Connected to 192.168.21.198.
Escape character is '^]'.
GET /bin/ps
<HTML><HEAD><TITLE>404 Not Found</TITLE></HEAD>
<BODY><H1>404 Not Found</H1>
The requested URL /bin/ps was not found on this server.
</BODY></HTML>
Connection closed by foreign host.
```

Il messaggio di allerta di Snort all'attacco web è il seguente:

```
[**] WEB-ATTACKS /bin/ps command attempt [**]
02/22-10:28:47.582594 192.168.21.193:1030 -> 192.168.21.198:80
TCP TTL:64 TOS:0x10 ID:52244 IpLen:20 DgmLen:65 DF
***AP*** Seq: 0x3DBAB39  Ack: 0xA0420F27  Win: 0x16D0  TcpLen: 32
TCP Options (3) => NOP NOP TS: 161595 33199
```

Che risponde alla seguente regola:

```
alert tcp any any -> 192.168.21.198 80
             (msg:"WEB-ATTACKS ps command attempt";
              uricontent:"/bin/ps"; nocase; sid:1328;
              rev:1; classtype:web-application-attack;)
```

che identifica un attacco atto a far eseguire a un server web mal configurato, l'applicazione `/bin/ps`.

Il secondo esempio mostra le reazione di snort all'utility nmap, un famoso portscanner. Con il seguente codice viene lanciato un nmap sulla macchina `192.168.21.198` e vengono visualizzati i servizi attivi con le relative porte.

```
root@192.168.21.193# nmap 192.168.21.198


Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.21.198):
(The 1593 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
22/tcp     open        ssh
80/tcp     open        http
111/tcp    open        sunrpc
443/tcp    open        https
3306/tcp   open        mysql
6000/tcp   open        X11
10000/tcp  open        snet-sensor-mgmt

Nmap run completed -- 1 IP address (1 host up) scanned in 2 second
```

Il messaggio di allerta di Snort al portscanning è il seguente:

```
[**] SNMP request tcp [**]
02/22-11:23:49.299340 192.168.21.193:47371 -> 192.168.21.198:161
TCP TTL:49 TOS:0x0 ID:44560 IpLen:20 DgmLen:40
******S* Seq: 0xFDDF7E00  Ack: 0x0  Win: 0x800  TcpLen: 2
```

che risponde alla seguente regola:

```
alert tcp any any -> 192.168.21.198 161 (msg:"SNMP request tcp";
flow:stateless; reference:bugtraq,4088; reference:bugtraq,4089;
reference:bugtraq,4132; reference:cve,2002-0012; reference:cve,2002-
0013; classtype:attempted-recon; sid:1418; rev:11;)
```

Questa regola ha come scopo quello di individuare la scansione della sola porta `161`, e viene quindi attivata dalla scansione di tutte le porte fatta da nmap.

3.1.4 Software utilizzati

Per poter utilizzare agevolmente ogni funzione di Snort ho installato e configurato i seguenti pacchetti.

3.1.4.1 Pacchetti da installare necessariamente

- *snort* -> IDS.

3.1.4.2 Pacchetti che si considerano già installati

- *apache* -> Web server;
- *mysql* -> Database relazionale;
- *php* -> Inteprete del linguaggio di programmazione omonimo;
- *Perl* -> Interprete del linguaggio di programmazione omonimo.

3.1.4.3 Pacchetti opzionali da installare

- *JPGraph* -> Librerie per la creazione di grafici tramite php;
- *ADODB* -> Librerie php per l'interfacciamento ai database;
- *Acid* -> Interfaccia web in php per l'elaborazione dei dati di snort;
- *Webmin* -> Interfaccia web in Perl per l'amministrazione di apache, mysql, snort e molto altro. Non ha bisogno di apache per funzionare.

3.1.4.4 Funzioni

Questi pacchetti servono a conservare (mysql), a consultare (apache, php, JPGraph, ADODB, Acid) e amministrare (Webmin, Perl) in maniera rapida ed efficace i report.

Lo schema di funzionamento finale dei vari pacchetti è il seguente:

Snort rileva i dati e li immette nel RDBMS MySQL, che viene amministrato tramite Webmin, i report inseriti nel database vengono elaborati e visualizzati tramite ACID che è scritto in php e fa uso di ADODB per accedere a MySQL e di JPGraph per creare i grafici. Acid ha bisogno di apache per funzionare, essendo un'applicazione web.

3.1.5 Procedimento eseguito per l'installazione dei pacchetti

3.1.5.1 MySQL

Per prima cosa occorre creare il database che verrà usato da snort:

```
shell> mysql –u root –p <digitare password>
```

una volta entrati nella shell di mysql creiamo il database "snort":

```
mysql> CREATE DATABASE snort;
```

successivamente ho attivato il database mysql per creare l'utente di accesso:

```
mysql> use mysql;
```

e ho dato i diritti all'utente snort:

```
mysql> GRANT INSERT,SELECT ON snort.* TO snort@127.0.0.1 \
IDENTIFIED BY 'password';

mysql> quit;
```

3.1.5.2 SNORT installazione

Si entra nella directory in cui si è scaricato snort e digitare da utente root:

```
rpm –Uvh snort*
```

alla fine si avrà snort istallato e già attivo, ma sono necessari vari interventi.

3.1.5.3 SNORT configurazione

Fermare il server snort tramite:

```
#/etc/init.d/snortd stop
```

per prima cosa occorre accertarsi che siano stati creati l'utente e il gruppo snort tramite lettura dei file `/etc/password` e `/etc/group`. Se non esistono occorre crearli con le apposite utility. Nel file `/etc/snort/snort.conf` occorre decommentare ed adattare alla rete la seguente variabile, ad esempio:

```
var HOME_NET 192.168.21.0/24
```

occorre controllare se il path delle regole è giusto:

```
var RULE_PATH /etc/snort/rules
```

Settare i dati di accesso a MySQL, decommentando questa riga output database:

```
log, mysql, user=snort password=<password> dbname=snort host=localhost
```

alla fine del file occorre commentare (# disattivare) o decommentare (attivare) le regole che ci servono.

Risulta molto utile attivare il rilevamento delle seguanti sezioni:

```
include $RULE_PATH/web-attacks.rules
include $RULE_PATH/shellcode.rules
```

È importante attivare anche il rilevamento di eventuali portscanner tipo "*nmap*":

```
preprocessor portscan: $HOME_NET 4 3 portscan.log
```

nel file `/etc/sysconfig/` occorre settare:

```
INTERFACE=eth0
CONF=/etc/snort/snort.conf
USER=snort
GROUP=snort
PASS_FIRST=0
LOGDIR=/va/log/snort
# ALERTMODE=fast
```

È fondamentale commentare, e quindi disattivare, il settaggio

“ALERTMODE” per utilizzare snort con mysql.

```
DUMP_APP=1
BINARY_LOG=1
NO_PACKET_LOG=0
```

Successivamente occorre popolare il database che userà snort:

```
# cd /usr/share/doc/snort*/contrib
# mysql –usnort –p<password> -Dsnort < create_mysql
# gzip –d snortdb-extra.gz
# mysql –usnort –p<password> -Dsnort < snortdb-extra
```

3.1.6 JPGraph

Queste librerie servono per potere creare grafici con ACID. Occorre scompattare il tarball direttamente nella Document Root di apache in `/var/www/html`.

```
# tar –xzvf jpgraph-1.14.tar.gz
```

Occorre quindi dare alle directory create `chmod 755` e ai files creati `chmod 644`.

3.1.7 ADODB

Queste librerie PHP servono a script tipo ACID per potere utilizzare vari tipi di database senza necessità di modificare il codice.

Occorre scompattare il tarball direttamente nella Document Root di apache.

```
# tar –xzvf  adodb411.tgz
```

Occorre quindi dare alle directory create `chmod 755` e ai files creati `chmod 644`.

3.1.8 ACID

È un’interfaccia per i report di snort scritta in php è veramente utile e

potente. Consente una consultazione molto chiara e organizzata dei dati, che altrimenti potrebbero risultare un ammasso indigeribile di stringhe.

Occorre scompattare anch'esso direttamente nella Document Root di apache.

```
# tar –xzvf  acid-0.9.6b23.tar.gz
```

Occorre quindi dare alle directory create `chmod 755` e ai files creati `chmod 644` ed effettuare i seguenti settagli in `acid/acid_conf.php`:

```
$DBlib_path = "/var/www/html/adodb";
$DBtype = "mysql";
$alert_dbname = "snort";
$alert_host= "localhost";
$alert_port = "";
$alert_user = "snort";
$alert_password = "<password>;
$archive_dbname = "snort";
$archive_host = "localhost";
$archive_port = "";
$archive_user = "snort";
$archive_password ="<password>";
$ChartLib_patch = "/var/www/html/jpgraph-1.14/src";
```

A questo punto occorre riavviare Snort con il seguente comando:

```
#/usr/sbin/snort –u snort –g snort –d –D –c /etc/snort/snort.conf
```

e verificare che apache sia attivo, con il seguente comando:

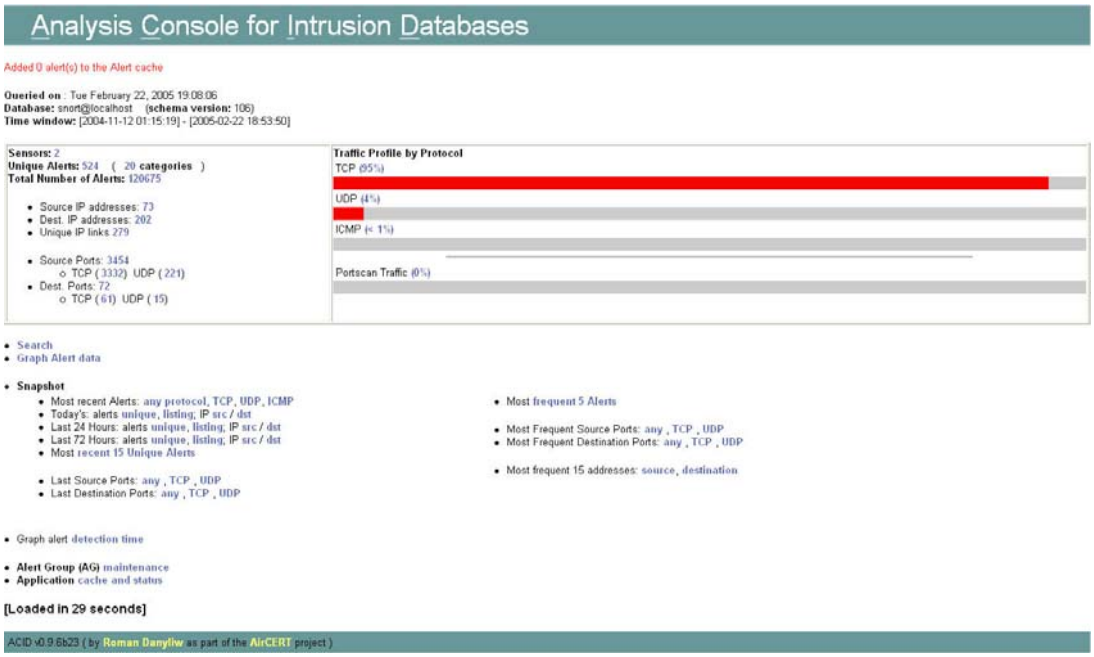
```
#/etc/init.d/httpd status
```

e si può accedere ad ACID tramite browser:

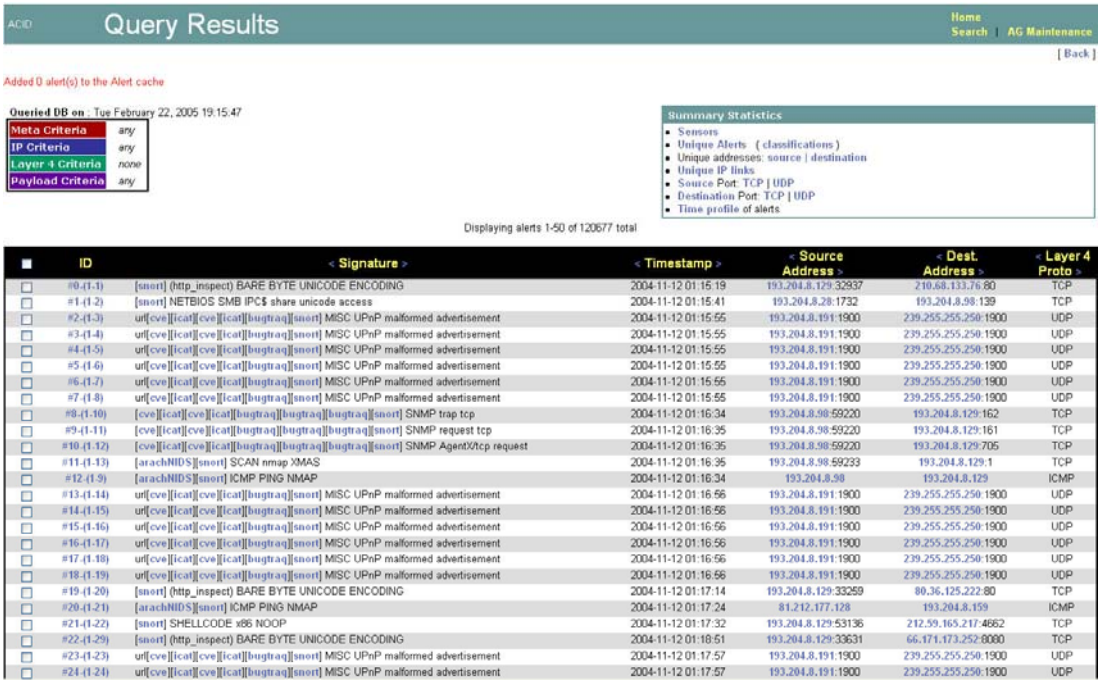
```
http://localhost/acid
```

al primo accesso acid chiederà di integrare il database con delle sue tabelle.

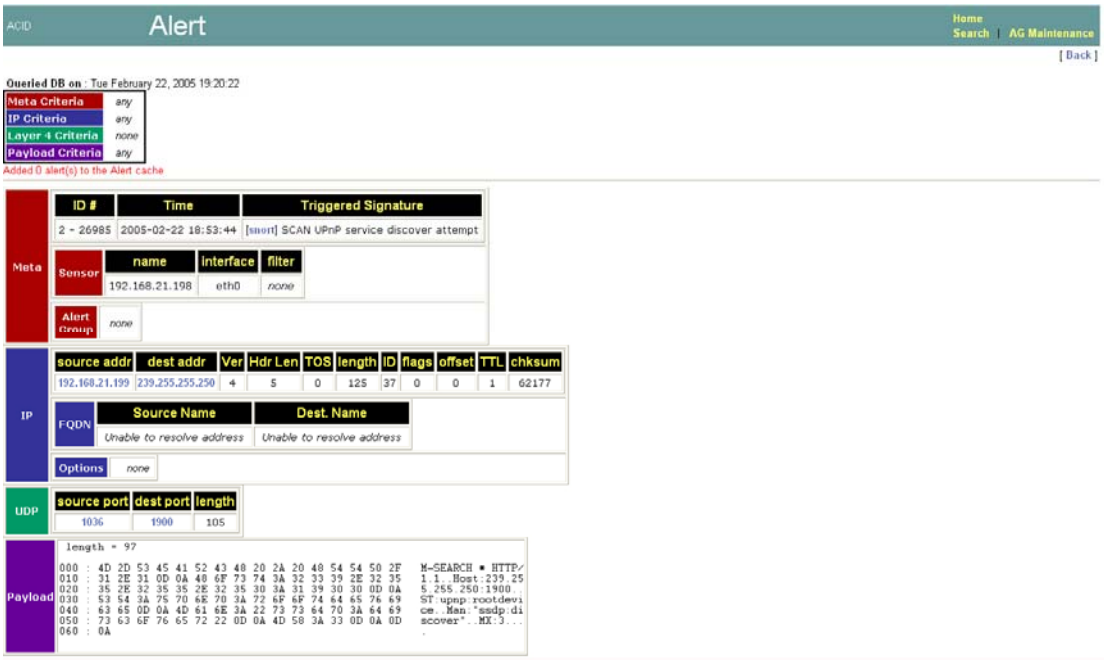
L'immagine successiva ci mostra il pannello di ingresso di Acid, dove si ha subito una prospettiva della situazione. In alto a sinistra sono indicati gli ultimi alert ancora non visionati. Tutte le scritte azzurre sono link a pagina di report elaborati nelle maniere più varie.



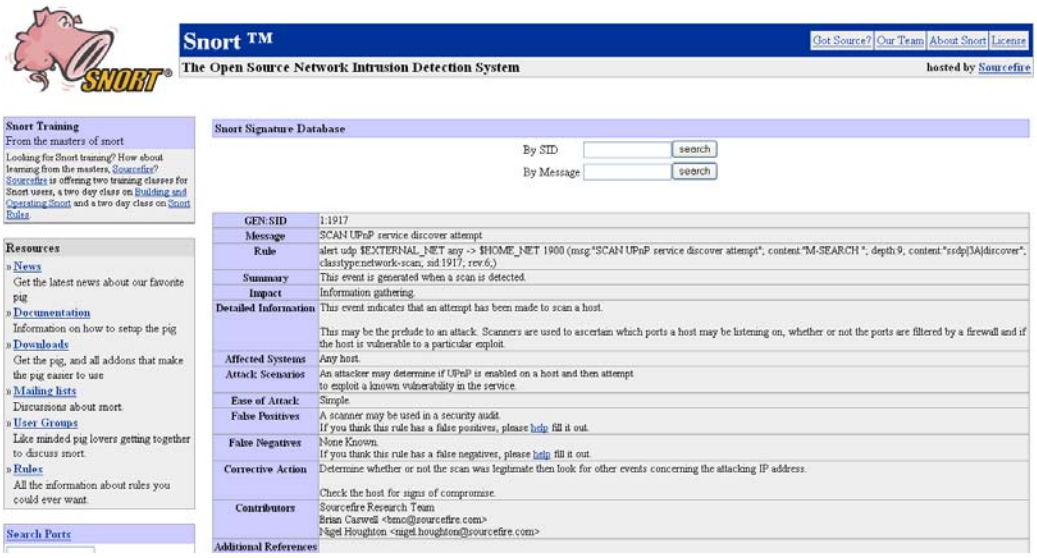
Questo è il pannello di report cronologico riassuntivo.



Questo è il pannello di approfondimento su un singolo alert, che si ottiene cliccando nel pannello precedente nei link nella colonna ID. Si tratta di tutte le informazioni possibili ottenibili tramite l'analisi dei pacchetti e eventuali riscontri in rete, come ad esempio DNS, ecc.



Qui è possibile vedere la pagina di spiegazione delle caratteristiche di un alert, comprensivo di livello pericolo, analisi tecnica, analisi di attacco, ecc. Si tratta di un servizio impagabile a cui si accede cliccando nei pannelli precedenti sui link “snort”.



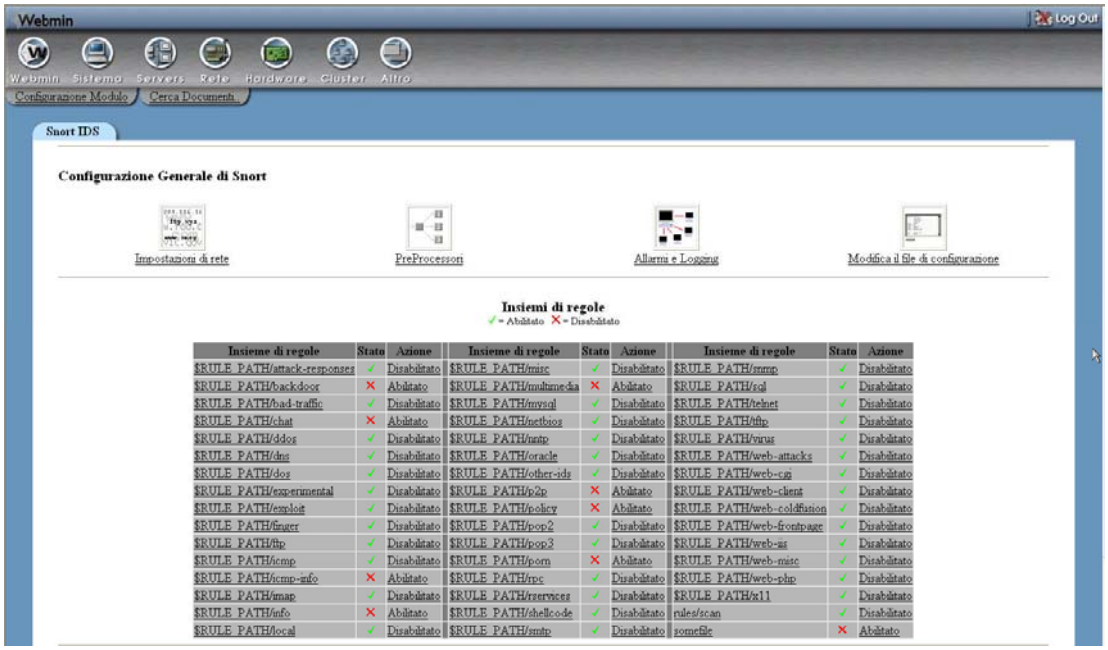
3.1.9 WEBMIN

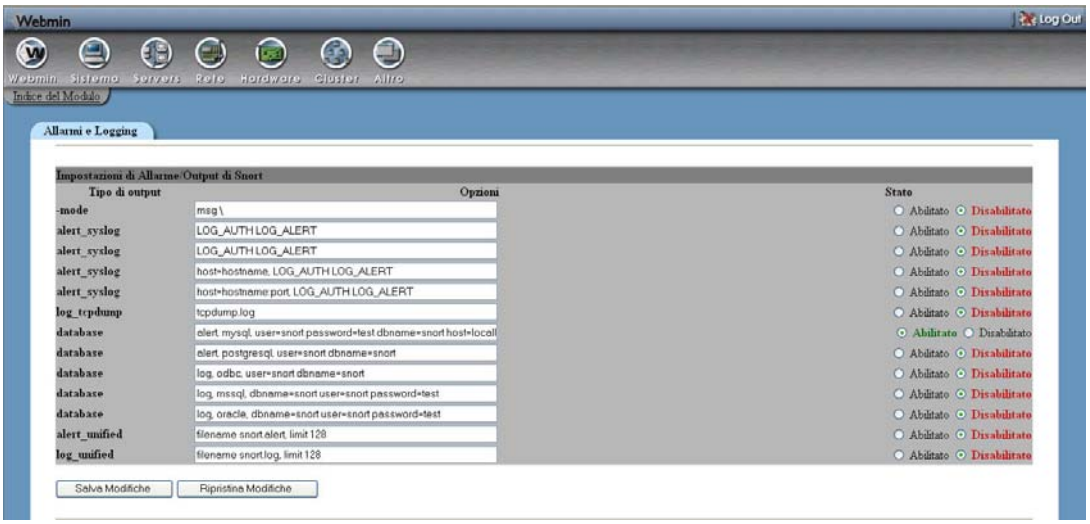
Si tratta di una potente interfaccia scritta in Perl. Ci si può fare di tutto,

gestire apache, mysql, snort, creare utenti, gestire interfacce di rete ecc. Una cosa importante è che pur essendo un applicazione web non ha bisogno di apache, in quanto è egli stesso un server, che gira sulla porta 10000.

La maggior parte dei moduli per gestire sono già istallati tramite rpm, alcuni, tra i quali snort, invece devono essere istallati successivamente.

Questi sono i pannelli di amministrazione delle regole di snort tramite Webmin che, dopo l'installazione è possibile trovare nella sezione "Server".





3.1.10 Modifiche

Nella versione istallata da me ho trovato un errore nel codice Perl del modulo che non mi consentiva di visualizzare, cliccando su "*Rule Set*", il contenuto della regola, cosa piuttosto fastidiosa.

Ho risolto inserendo nel file*/usr/libexec/webmin/snort/index.cgi* alla riga 77, la seguente stringa:

```
$rule =~ s/\$RULE_PATH//;
```

che non fa altro che eliminare *"$RULE_PATH*" dall'URL linkato nella pagina che per qualche motivo non viene interpolata [22].

3.2 Guardian

Snort è soltanto un'IDS passivo, ossia in grado di rilevare delle eventuali intrusioni, ma non provvede a bloccarle. Quindi per ovviare a questa mancanza si

può installare ed utilizzare Guardian.

Guardian (www.chaotic.org/guardian/) è un programma di sicurezza che funziona automaticamente insieme a Snort per aggiornare le regole del firewall, basate sugli allarmi generati da Snort. Le regole aggiornate del firewall bloccano tutti i dati ricevuti dal IP ADDRESS della macchina d'attacco ossia la macchina che ha indotto Snort a generare un allarme.

È provvisto anche di metodi che fanno in modo che gli IP ADDRESS del DNS, Gateway, ecc. non vengano bloccati.

Innanzitutto Guardian è uno script in Perl che bisogna far partire tramite shell:

```
/usr/local/bin/guardian.pl –c /etc/guardian.conf
```

Questo comando deve essere impartito sulla shell, dopo aver avviato snort, con il seguente comando:

```
/usr/sbin/snort/ -A fast –b –d –D –i eth0 –u snort –g snort

–c /etc/snort/snort.conf -l /var/log/snort
```

non fa altro che avviare Guardian in backgroun, secondo le impostazioni che sono presenti nel proprio file di configurazione `/etc/guardian.conf`, che sono le seguenti:

```
Interface       eth0
```

Interfaccia in cui Guardian si mette in ascolto;

```
HostGatewayByte 1
```

L'ultimo ottetto del IP address, che ci dà l'indirizzo del Gateway;

```
LogFile         /var/log/guardian.log
```

File di log in cui vengono riportate le informazioni di Guardian.

```
AlertFile       /var/log/snort/alert
```

File di alert in cui Snort, tramite il syslog, va ad inserire le informazioni delle possibili minacce.

Guardian non fa altro che controllare le nuove informazioni inserite, per poter attivare delle regole sul firewall in grado di bloccare IP ADDRESS di un'eventuale malintenzionato.

```
IgnoreFile      /etc/guardian.ignore
```

Nel seguente file invece vengono ipostati gli indirizzi IP delle macchine che non devono assolutamente essere bloccate, come il DNS, il Gateway ecc.

```
TargetFile      /etc/guardian.target
```

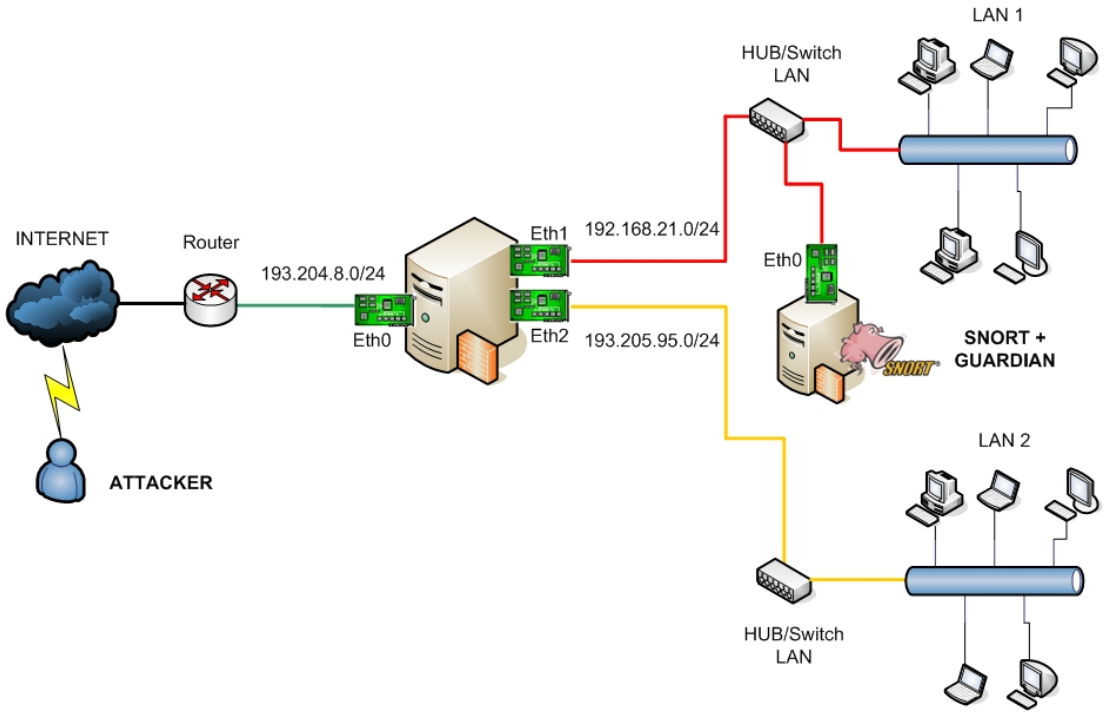
In questo file invece vengono impostati gli altri eventuali indirizzi IP dell'host.

```
TimeLimit   3600
```

Con questa impostazione invece si stabilisce il tempo massimo per cui la regola impostata sul firewall, deve durare.

3.2.1 Funzionamento di Guardian

Nell'immagine sottostante viene mostrato come Snort e Guardian lavorano sulla stessa macchina, e interagiscono tramite la rete, con il firewall presente all'indirizzo IP `192.168.21.1`, inserendo nuove regole e togliendole dopo un tempo prestabilito.



Il funzionamento del sistema è il seguente: l'IDS Snort è in ascolto sulla rete interna `192.168.21.0/24` quindi memorizza, tramite il syslog, nel file `/var/log/snort/alert`, le informazioni relative ad eventuali possibili intrusioni nella LAN. Il formato di salvataggio nel file alert è il seguente:

```
02/23-11:20:18.424259  [**] [1:1420:11] SNMP trap tcp [**]
[Classification: Attempted Information Leak] [Priority: 2] {TCP}
192.168.21.196:49212 -> 192.168.21.198:162
```

In cui risulta evidente la data e l'ora, il tipo di attacco, l'indirizzo IP dell'eventuale malintenzionato, con la relativa porta, e l'indirizzo IP della macchina bersaglio, con la relativa porta.

Guardian controlla costantemente l'inserimento di nuovi messaggi all'interno del log e preleva da essi l'indirzzo IP del malintenzionato. Successivamente tramite ssh invia, attraverso la rete `192.168.21.0/24`, alla macchina in cui è presente il firewall, le regole che andranno a droppare tutto il traffico proveniente dal malintenzionato.

Per poter inviare automaticamente delle regole per il firewall occorre che ci sia un collegamento ssh con chiave crittografica tra i due computer, in modo che la macchina in cui è presente Guardian, possa effettuare il login, sulla macchina con il firewall, senza dover digitare la password. Il codice sottostante ci mostra come creare le chiavi pubblica e privata di tipo dsa:

```
ssh-keygen -b 1024 -t dsa

Generating public/private dsa key pair.

Enter file in which to save the key (/home/mario/.ssh/id_dsa): Created
directory '/home/mario/.ssh'.
```

Viene chiesto di inserire una passphrase per le chiavi, per motivi di sicurezza è bene usufruirne

```
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
```

Viene creato la chiave privata che dovrà rimanere sulla macchina locale

```
Your identification has been saved in /home/mario/.ssh/id_dsa.
```

Viene creato il file contenente la chiave pubblica che dovrà essere presente anche sul server remoto [23].

```
Your public key has been saved in /home/mario/.ssh/id_dsa.pub.
The key fingerprint is:
31:ec:ee:82:52:2f:3e:46:6f:18:97:9d:3a:a5:7b:dc mario@dido
```

Gli script, che Guardian richiama, per poter inserire nuove regole al firewall sono i seguenti:

```
source=$1
interface=$2
firewall_ip="192.168.21.1"
```

```
ssh -vvv root@$firewall_ip "/sbin/iptables -I INPUT -s $source -i
$interface -j DROP"

ssh -vvv root@$firewall_ip "/sbin/iptables -I FORWARD -s $source -d
193.205.95.0/24  -j DROP"

ssh -vvv root@$firewall_ip "/sbin/iptables -I FORWARD -s $source -d
192.168.21.0/24  -j DROP"
```

Il codice sovrastante fa parte dello script *guardian_block.sh*, da me realizzato, in cui sono visibili le regole che verranno applicate automaticamente al firewall.

```
source=$1
interface=$2
firewall_ip="192.168.21.1"

ssh -vvv root@$firewall_ip "/sbin/iptables -D INPUT -s $source -i
$interface -j DROP"

ssh -vvv root@$firewall_ip "/sbin/iptables -D FORWARD -s $source -d
193.205.95.0/24 -j DROP"

ssh -vvv root@$firewall_ip "/sbin/iptables -D FORWARD -s $source -d
192.168.21.0/24 -j DROP"
```

Il codice sovrastante fa parte dello script *guardian_unblock.sh*, in cui sono visibili le regole che verranno tolte dal firewall dopo un tempo prestabilito.

3.2.2 Esempio di funzionamento

Nel seguente paragrafo riporto un esempio del funzionamento di Snort con Guardian.

Innanzitutto ho eseguito un portscanning con NMAP alla macchina con installato Snort e Guardian, da una macchina all'interno della rete LAN. Il computer da cui ho lanciato il portscanning, aveva il seguente indirizzo IP `192.168.21.196`:

```
192.168.21.196# nmap –sS –O 192.168.21.198
```

che fornisce il seguente risultato:

```
Starting nmap V. 3.00 ( www.insecure.org/nmap/ )
Interesting ports on  (192.168.21.198):
(The 1593 ports scanned but not shown below are in state: closed)
Port       State       Service
21/tcp     open        ftp
22/tcp     open        ssh
80/tcp     open        http
111/tcp    open        sunrpc
443/tcp    open        https
3306/tcp   open        mysql
6000/tcp   open        X11
10000/tcp  open        snet-sensor-mgmt
Remote operating system guess: Linux Kernel 2.4.0 - 2.5.20
Uptime 0.008 days (since Wed Feb 23 17:15:35 2005)

Nmap run completed -- 1 IP address (1 host up) scanned in 2 seconds
```

Come risulta evidente dal codice successivo, dopo il portscanning, è stata aggiunta la regola, che blocca qualsiasi pacchetto proveniente dall'indirizzo IP del malintenzionato (`192.168.21.169`), sul firewall IPTables.

```
192.168.21.1 #/sbin/iptables -L

Chain INPUT (policy DROP)
target     prot opt source               destination
DROP       all  --  192.168.21.196       anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination
DROP       all  --  192.168.21.196       192.168.21.0/24
DROP       all  --  192.168.21.196       193.205.95.0/24
```

3.3 P2P WatchDog

P2P WatchDog (www.p2pwatchdog.com/) è destinata per controllare l'attività dei programmi peer-to-peer di file-sharing che ripartisce l'attività sulle reti di grande capacità. Attualmente, i seguenti client e protocolli possono essere rintracciati senza tenere conto della porta che stanno utilizzando:

- BearShare;

- BitTorrent;
- Blubster;
- DC++;
- DirectConnect;
- Earthstation 5 (SSL);
- eDonkey;
- eMule;
- FastTrack;
- Filetopia (SSL);
- Gnucleus;
- Gnutella;
- Grokster;
- iMesh;
- KaZaA;
- Kazaa Lite;
- LimeWire;
- Morpheus;
- MP2P;
- Overnet;
- Piolet;
- RockitNet;
- Shareaza;
- WinMX;
- XoloX.

In molti casi, i client possono anche essere rintracciati se stanno usando un proxy HTTP di terzi o client che usano gli schemi di crittografia quale SSL, persino passanti all'interno di proxy server.

Quando uno di questi protocolli è usato per trasferire file o dati, le seguenti informazioni vengono registrate:

- Data e tempo del trasferimento;
- Indirizzo IP e porta utilizzata;
- Indirizzo MAC hardware per connessioni non PPP;
- Nome, autore, dimensioni del file, e tipo di contesto (secondo il client che ripartisce il file, non disponibile in tutti i casi).

Supporta i seguenti adattatori di rete:

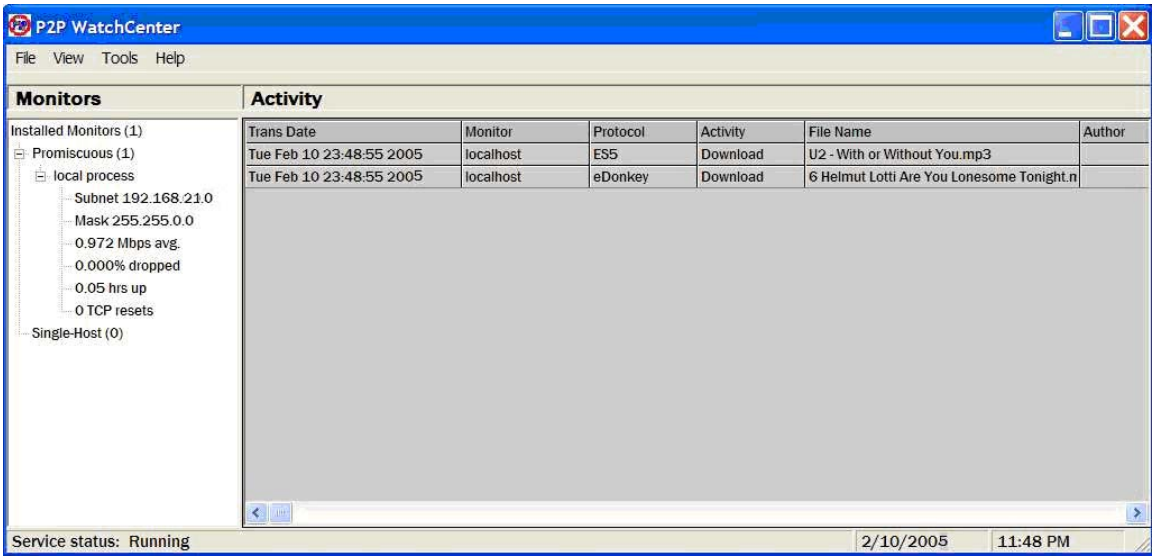
- Connessione Ethernet diretta (10/100/1000 Mbps);
- Reti Wireless 802.11;
- Connessioni Cable e DSL;
- Connessioni Dialup PPP.

Soltanto una delle macchine sulla rete può controllare il traffico sul segmento della rete. Dopo l'installazione dell'applicazione il WatchCenter funzionerà come servizio in background e sarà visibile nel Task Manager. Il server si avvierà automaticamente ogni volta che il calcolatore si riavvia.

3.3.1 Monitorare l'attività

Per default, quando un programma P2P viene attivato, l'assistente genererà un messaggio pop-up che contiene gli indirizzi della fonte e delle destinazione. Le informazioni dettagliate sull'attività dei file in upload e download, sono disponibili con l'applicazione di P2P WatchCenter. Se più di 10 dialoghi pop-up vengono contemporaneamente visualizzati, i dialoghi supplementari non sono generati. Tuttavia, l'attività di file-sharing sarà monitorata e registrata. Quando si lancia il server, i file collegati vengono elencati a sinistra dal IP ADDRESS della macchina. Se c'è un file installato sulla stessa macchina del server, allora sia l'interfaccia di "loopback" (127,0,0,1) che l'adattatore della rete verranno elencati. L'esposizione dell'attività include soltanto i 150-200 trasferimenti più recenti da quando il server è stato avviato.

La seguente immagine mostra il pannello di controllo dell'applicazione:



Come è evidente lo schermo è diviso in due sezioni. Il pannello di sinistra elenca tutti i file P2P attualmente rilevati dal server. Il numero totale di file P2P compare nella prima fila. I file sono classificati in base al loro IP ADDRESS, o "dal processo locale" se il file sta funzionando sullo stesso computer che ospita il server WatchCenter.

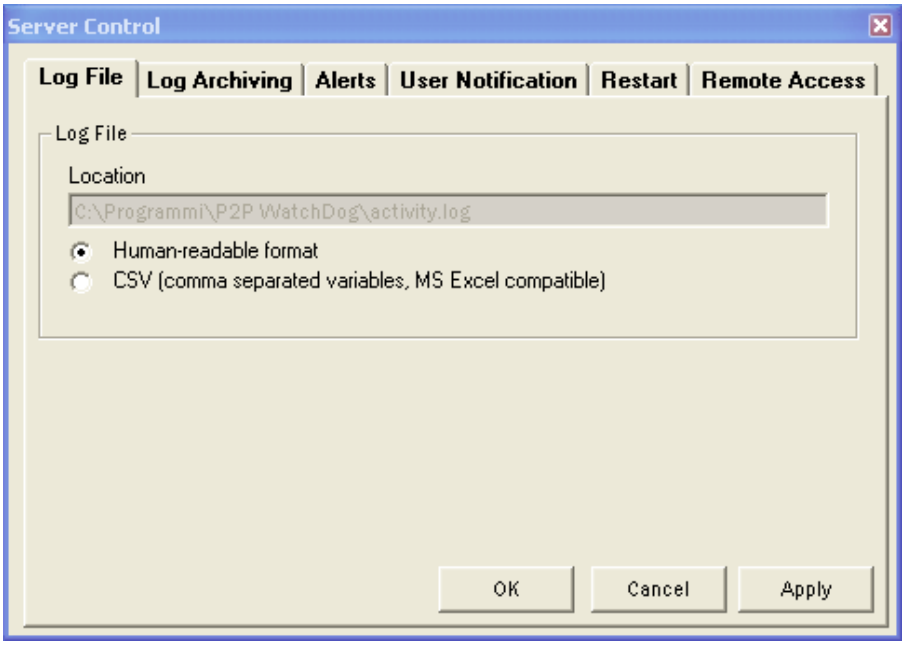
Il pannello di destra contiene una lista di attività recenti, nelle ultime 100 ore, o dall'ultima volta che l'esposizione è stata eliminata, o da quando il server è stato avviato, per l'ultima volta. La capienza massima dell'esposizione è 200 file e se questo numero è oltrepassato, alcuna dell'attività non sono visualizzata. Le intestazioni della colonna sono:

- *Trans Date* - La data ed il tempo in cui il trasferimento è stato rilevato;
- *Monitor* - l'IP ADDRESS del monitor che ha rilevato l'attività, o "0,0,0,0" se il monitor è installato sulla macchina locale;
- *Protocol* - Il protocollo usato per il trasferimento. (nell'esempio qui sopra ES5/eDonkey);
- *Activity* - Upload / download / connection / remote connection;

- *File Name* - Il nome della file. In alcuni casi questo non può essere segnalato, se è cifrato, o la richiesta è basato su un file hash;
- *Author* - Alcuni client segnalano il nome dell'autore;
- *Type* - il formato del file;
- *Bytes* - La dimensione del file richiesto. Questi non sono i byte realmente trasferiti, poiché il trasferimento può essere bloccato;
- *Int IP* - l'IP ADDRESS dell'host interno alla rete, con il nome se la risoluzione di Domain Name è stata impostata;
- *Ext IP* - I'IP ADDRESS dell'host esterno, o il nome se la risoluzione di Domain Name è stata impostata;
- *Proxy IP* - Se si sta utilizzando un proxy, mostra l'IP ADDRESS del proxy;
- *Orig MAC* - Il MAC ADDRESS del creatore.

Comunque è possibile ciccare sopra l'header della colonna per poter ordinare in base alla contenuto della colonna.

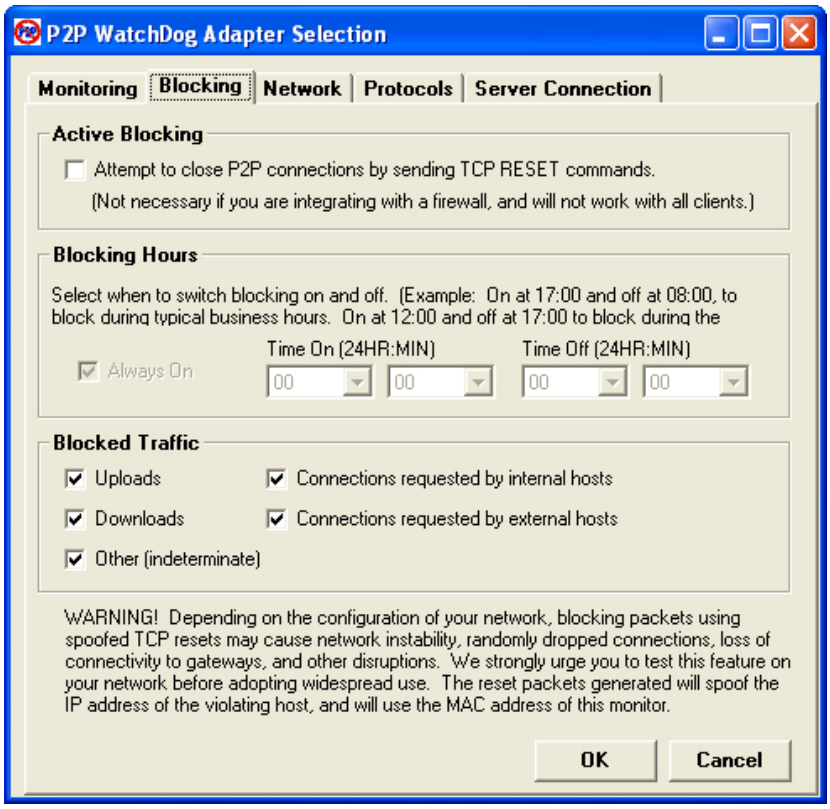
Nell'immagine successiva invece è possibile osservare il pannello per il controllo del server:



Che contiene i seguenti pannelli:

- *Log File*: per impostare la posizione del file di log;
- *Log Archivino*: in cui è possibile impostare la dimensione massima del file dei log, o se memorizzarlo periodicamente;
- *Activity Alerts*: è possibile impostare come deve essere avvertito l'amministratore della presenza di utenti che utilizzano programmi p2p;
- *User Notification*: è possibile inviare dei messaggi agli utenti che stanno per utilizzare programmi p2p;
- *Restart*: da la possibilità di far ripartire l'applicazione se si verifica un errore.

Nell'immagine successiva invece è possibile osservare il pannello (Configure Monitor) per il controllo del Monitor:



Che contiene i seguenti pannelli:

- *Monitoring*: per impostare le interfacce su cui il monitor si deve mettere in ascolto;
- *Blocking*: in cui è possibile impostare se bloccare le connessione p2p, l'intervallo di tempo in cui bloccarle, e il tipo (Download, Upload, ecc.);
- *Network*: in cui è possibile impostare l'indirizzo e la maschera di rete in cui il monitor si deve mettere in ascolto;
- *Protocols*: in cui è possibile impostare i tipi di protocolli p2p da bloccare;
- *Server Connection*: in cui è possibile impostare l'indirizzo IP in cui gira il server, nel caso in cui non è installato sulla stessa macchina del client [24].

Comunque nonostante tale programma riesca ad individuare e bloccare parecchi tipi di protocolli e clients p2p, non riesce comunque ad individuarli e bloccarli tutti. Infatti tutti i clients che usano la crittografia, per inviare e ricevere dati, possono operare tranquillamente. I client non rilevabili che ho potuto provare sono i seguenti:

- *ANts P2P:* è un software peer-to-peer di terza generazione che protegge la privacy mentre si è connessi ad internet e rende irrintracciabile, nascondendo l'identità e criptando tutti i dati che si invia o si riceve;
- *FreeNet:* è un software che permette di pubblicare e reperire file ed informazioni da internet senza il timore di venire censurato. La rete è completamente decentralizzata. Sia chi pubblica file e notizie e sia chi legge o scarica, conserva un completo anonimato;
- *MUTE:* protegge la privacy evitando le connessioni dirette tra utenti che vogliono scambiarsi file. La maggior parte dei programmi peer to

peer usano infatti connessioni dirette per scaricare o uploadare file e questo ovviamente rende facile tramite l’intercettazione dell'indirizzo IP scoprire l’identità degli utenti [25].

CAPITOLO 4

TRAFFIC SHAPING

4.0 Introduzione al Traffic Shaping

Traffic Shaping è un generico termine assegnato ad una moltitudine di tecniche atte a sviluppare politiche di prioritizzazione del traffico di dati in una rete. La maggior parte degli utenti di una rete hanno, prima o poi, sperimentato la frustrante esperienza della latenza e accodamento dei dati della rete internet.

Chi ha usato servizi come telnet o ssh su un collegamento lento come un dial-up PPP o SLIP, ha visto gli effetti che il file transfer ha sul traffico interattivo. Il file transfer (download o upload che sia) divora facilmente le risorse di larghezza di banda costringendo il servizio interattivo ad un lungo accodamento in attesa di uno slot libero prima di poter trasmettere.

Il problema è causato per il fatto che il flusso dei dati del file transfer ha uguale priorità rispetto a quello del servizio interattivo. Il traffico è regolato da una coda FIFO (First In, First Out).

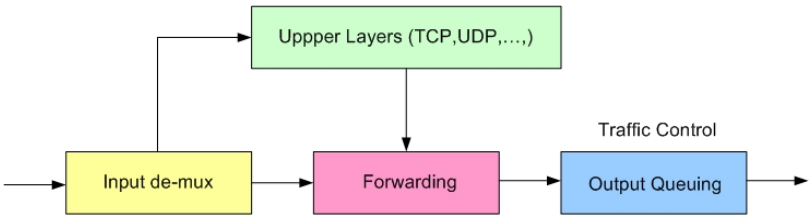
Il Traffic Shaping ci permette di implementare una specifica politica che altera il meccanismo secondo cui vengono accodati i dati in trasmissione.

È necessario a questo punto chiarire che una politica di shaping può essere applicata solo al traffico in uscita da un'interfaccia. Il traffico in arrivo, nel momento in cui è giunto all'interfaccia, ha ormai irrimediabilmente occupato una certa banda di trasmissione e non è più possibile operare alcun incodamento straordinario.

Questo però non significa che sia impossibile modellare il traffico in download verso la nostra interfaccia.

4.1 La struttura del kernel e i comandi per il controllo del traffico

Linux offre un ricco set di funzioni per il controllo del traffico come i comandi *tc*, *iptables*, *iproute* e così via i quali si appoggiano su una struttura definita nel kernel del sistema operativo. Cerchiamo di definire in che modo vengono elaborati i pacchetti all'interno delle macchine linux. A livello più alto possibile possiamo immaginare una situazione come quella di figura:



La figura mostra come il kernel processi i dati ricevuti dalla rete e come esso possa generare nuovi dati che dovranno essere spediti sulla rete: i pacchetti entranti possono sia essere reindirizzati immediatamente sulla rete (magari su diverese interfacce nel caso di un router), sia essere passati al livello superiore della pila protocollare (come ad esempio un protocollo di trasporto) per essere processati.

Il forwarding include la selezione dell'interfaccia di uscita, la selezione del prossimo salto, l'incapsulamento e così via. Una volta fatto questo i pacchetti sono accodati sulle rispettive interfacce dove vigeranno le discipline di accodamento per la gestione del traffico.

4.2 Iproute2

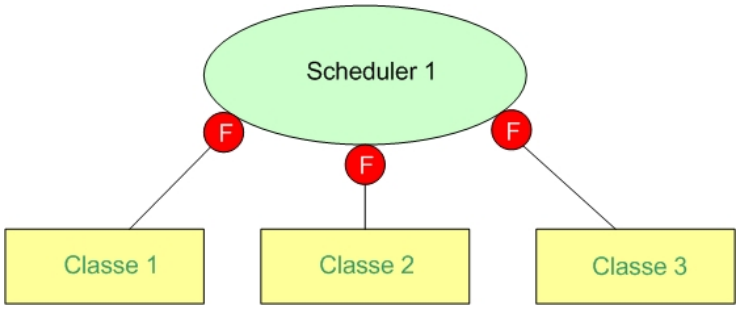
All'interno del blocco input de-multiplex possiamo immaginare la routing table relativa ai pacchetti provenienti dall'esterno e che devono essere reindirizzati. Il pacchetto per la gestione dell'indirizzamento è iproute2.

Questo risulta essere uno strumento molto flessibile e completo in quanto è possibile definire più tavole di routing da consultare a seconda della sorgente del pacchetto, della destinazione o semplicemente dell'utente che richiede l'utilizzazione della risorsa. Tra la miriade di comandi disponibili quelli di cui ha più senso parlare in questa sede sono:

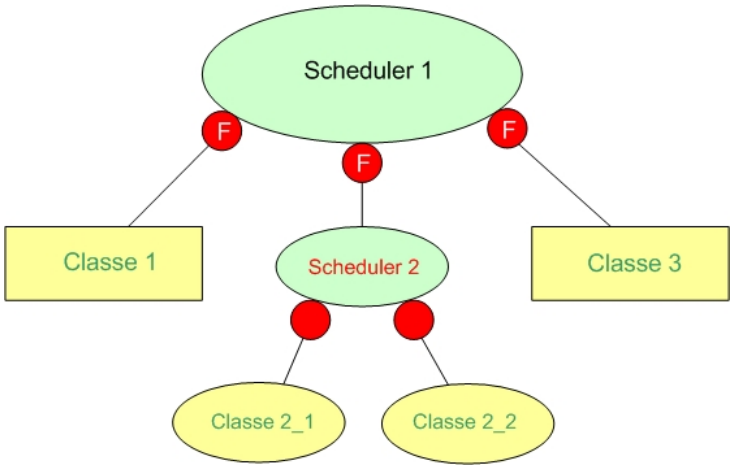
- *ip route*: aggiunge o cancella una riga della routing table in cui sono contenute le informazioni sui percorsi per raggiungere gli altri nodi di rete;
- *ip rule*: definisce delle regole di instradamento non dipendenti solo dalla sorgente e dalla destinazione del pacchetto ma anche da altri campi dell'header. Per fare ciò definisce più routing table a seconda del tipo di pacchetto. Ma le diverse routing table dovranno comunque essere gestite tramite ip route;
- *ip link*: setta le caratteristiche dei dispositivi di uscita (schede di rete, modem);
- *ip address*: associa indirizzi ip ai dispositivi di rete.

Internamente ad iproute, il concetto di classe è utilizzato per aggregare traffico con caratteristiche comuni. Si hanno quindi i filtri che servono a determinare quali pacchetti devono andare in ogni determinata classe. Avendo poi pacchetti presenti in più classi, è compito dello scheduler decidere in che ordine servire le diverse classi.

Una delle caratteristiche più interessanti di iproute è la possibilità di avere una gerarchia di elementi. Supponiamo ad esempio di avere una configurazione come nella figura sottostante:



Se ad un certo punto ci si rende conto che all'interno di una certa classe ci sono diversi sottotipi di traffico che necessitano di essere trattati diversamente, è possibile sostituire la classe con uno scheduler sotto al quale porre diverse sottoclassi e i rispettivi filtri. Come risulta evidente nell'immagine sottostante.



Non è poi obbligatorio utilizzare gli stessi algoritmi di incodamento (queueing) per le classi di uno stesso scheduler.

Sempre nella figura sovrastante si potrebbe ad esempio utilizzare la tecnica

FIFO per la classe1, RED per la classe2 e via dicendo. Questo consente di avere un alto grado di flessibilità.

4.3 Queueing discipline

La queueing discipline di iproute altro non sono che schedulers. Nella tabella sottostante sono riportate le componenti di iproute divise per categoria. Non potendole vedere tutte in dettaglio, spiegherò il funzionamento delle più importanti.

Schedulers a più classi	
CBQ	(Class-Based Queueing)
CSZ	(Clark-Shenker-Zhang)
ATM	(Asynchronous Transfer Mode)
PRIO	(Priority)
Gestione code (Scheduler a classe unica)	
RED	(Random Early Detection)
GRED	(Generalized RED)
SFQ	(Stochastic Fair Queuing)
TBF	(Token Bucket Flow)
TEQL	(Traffic Equalizer)
PFIFO	(Paket FIFO)
BFIFO	(Byte FIFO)
Classificatori	
Routing table based	
Firewall based (es. iptables)	
U32	
RSVP/RSVP32	
Cls_rt	
Tc	

4.3.1 PRIO

Questo scheduler semplicemente mantiene più classi a diversa priorità e le serve in ordine decrescente di quest'ultimo. Quando si utilizza questo tipo di scheduler, occorrerebbe assicurarsi che non sia possibile che il traffico a minor priorità possa soffrire di "*starving*", ossia rimanga in attesa indefinita a causa della permanente presenza di traffico prioritario.

Quando questo accade significa probabilmente che l'assegnazione della priorità non è stata fatta correttamente. Una soluzione consiste ovviamente nel ridefinire le classi e i filtri (quale tipo di traffico in quale classe) o agganciare a una o più di queste una tecnica di incodamento che ne limiti la dimensione. In questo modo le classi prioritarie mantengono la precedenza sulle altre, ma, avendo un rate massimo associato, assicurano alle rimanenti di ricevere una parte della banda.

4.3.2 CBQ (Class Based Queueing)

CBQ [27] rappresenta sicuramente lo scheduler più complicato tra quelli disponibili per Linux. I suoi principi consistono nella possibilità di ripartizionare la banda tra protocolli diversi e/o tra diversi utenti (o organizzazioni). Una loro combinazione (ripartizione sia per utenti che per protocolli) è ovviamente ancora possibile (l'algoritmo è gerarchico e consente di ripartire a piacere ciascuna sottoclasse). Ci sono comunque altre interessanti opzioni, che però ne complicano l'implementazione.

Quando una delle classi non utilizza completamente la fetta di banda assegnata, i surplus potrebbe risultare uno spreco se riciclato opportunamente dalle altre classi, in particolare da quelle che si trovano ad avere bisogno di più banda di quella a loro disposizione.

Per ogni classe è inoltre possibile definire i seguenti flag:

- *Bounded*: La classe non può prendere in prestito alcuna banda dai suoi antenati. Se ad esempio classe 2 e classe 3 non hanno traffico, scheduler 1 si trova ad avere della banda a disposizione che è inutilizzata. Se dichiariamo classe 1 bounded, significa che questa non può usufruire di tale surplus;

- *Isolated*: La classe non può né prendere in prestito dagli antenati né prestare a classi non sue discendenti. Non solo classe 1 non può usare la banda di scheduler non utilizzata da classe 2 e 3, ma, nel caso classe 1 fosse dichiarata isolated, la banda assegnata a classe 1 che non viene usata non potrebbe essere riciclata da classe 2 e 3.

4.4 Algoritmi di incodamento

Uno schedaler a senso quando sono presenti più code. In presenza di un'unica coda, non si para più di scheduler, ma di strategia di incodamento. In realtà scheduler e strategia di incodamento sono la stessa cosa e differiscono solo nel numero di code (classi nel nostro caso) che gestiscono.

Dopo tutto una tecnica di incodamento riceve elementi secondo una certa sequenza temporale e lì preleva con un'altra sequenza o, nel caso particolare della coda FIFO, nella stessa sequenza.

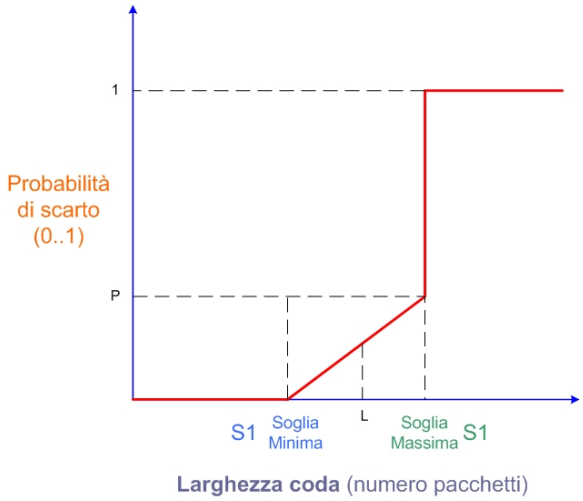
Al contrario della queueing disciplines, gli algoritmi di incodamento quindi lavorano (normalmente) su una singola classe. Essendo una sola la classe gestita da questi scheduler, coda o buffer è probabilmente un nome più appropriato.

4.4.1 CPFIFO/BFIFO (Packet FIFO/Byte FIFO)

Questi due scheduler rappresentano due semplici varianti della versione base FIFO. L'unica differenza consiste nel modo in cui viene specificata la lunghezza della coda. PFIFO definisce la dimensione della coda in termini di "numero pacchetti" mente BFIFO in termini di byte. In alcuni casi, ad esempio può essere più comodo definire la coda di lunghezza 100 pacchetti; in altri contesti invece può essere meglio dire che non può contenere più di 100 Kbyte.

4.4.2 RED (Random Early Detection)

Il Grafico di figura mostra come varia la probabilità che un nuovo pacchetto venga posto in coda o scartato a seconda del valore di alcuni parametri.



Una coda RED [28] può essere vista come una variante particolare di una coda FIFO. Mente per FIFO occorre specificare solo la dimensione massima della coda e tutti i pacchetti che arrivano dopo che la coda è piena sono scartati incondizionatamente, i RED cambia il principio utilizzato per decidere se accettare un nuovo pacchetto (metterlo in coda) o scartarlo. I pacchetti sono comunque estratti secondo la logica FIFO. Quello che cambia quindi è solo il principio di inserimento.

Anzitutto occorre definire due sogli anziché una: soglia minima e soglia massima. Serve inoltre specificare la probabilità con cui scartare i pacchetti.

Per semplicità chiamiamo questi pacchetti S1, S2, e P, come è possibile vedere nella figura sovrastante. Fin tanto che la lunghezza della coda si mantiene al di sotto di S1, i pacchetti sono aggiunti alla coda e questa si comporta in tutto e per tutto come una FIFO.

Quando la coda ha una lunghezza maggiore o uguale a S2 i pacchetti sono

scartati incondizionatamente, il che significa che la coda non potrà mai superare la lunghezza S2.

Tra S1 e S2 invece, la probabilità che un pacchetto venga scartato, e quindi non inserito in coda varia linearmente tra 0 (in S1) e P (S2). Supponendo di avere le due soglie con valori S1=50, S2=100 e P=25%, la tabella riporta la probabilità in corrispondenza di diversi valori.

Elementi in Coda	Probabilità di scarto (0..1)
50 (S1)	0
55	0.025
60	0.05
65	0.075
70	0.1
75 (P)	0.125
80	0.15
85	0.175
90	0.2
95	0.225
100 (S2)	0.25
101	1

GRED rappresenta una variante di RED, che consente di avere più classi di priorità su cui poter mappare diversi livelli di priorità di scarto. Per default le classi sono 16, ma il numero può essere cambiato.

4.4.3 TBF (Token Bucket Flow)

Il token bucket viene normalmente utilizzato per limitare la banda di una classe di traffico. Sebbene CBQ e BFIFO consentano apparentemente di ottenere la stessa cosa, c'è in realtà una sottile differenza.

Il principio di funzionamento è molto semplice. Supponiamo di voler forzare una banda a 1 Mbit/s. Fissiamo un numero "n" a piacere ogni 1/n di secondo viene incrementata la banda a disposizione di 1/n Mbits. Questo consente di distribuire la banda a disposizione uniformemente nell'unità di secondo. Se ad esempio n valesse 100, allora ogni 1/100 di secondo verrebbe aggiunto 1/100 Mbits = 10 Kbits alla

banda a disposizione, stando ovviamente sempre attenti a non superare il valore massimo (nell'esempio 1Mbits).

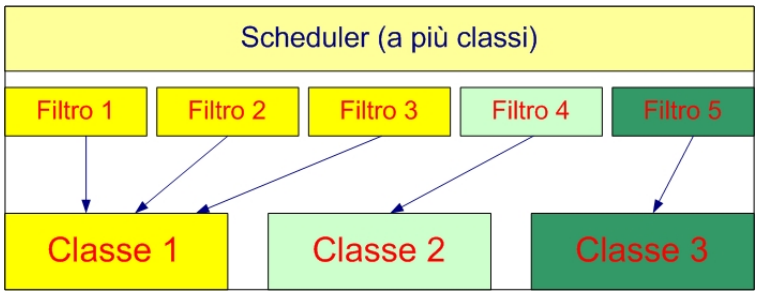
Il comportamento da seguire nel caso non ci fosse banda a disposizione dipende dalla configurazione dello scheduler. Normalmente il traffico in eccesso viene scartato. Una possibile variante consiste nel declassare il traffico, ovvero penalizzarlo riducendogli la priorità.

4.5 Classificatori

Un classificatore altro non è che una coppia (condizione, classe) che associa il traffico che soddisfa la condizione alla classe.

Ovviamente più filtri possono essere associati ad una stessa classe. Per ovvie ragioni non ha senso una classe senza alcun filtro associato.

I classificatori si differenziano per il tipo di condizioni che consentono di esprimere, sebbene sia talvolta possibile esprimere lo stesso vincolo con diversi classificatori.



4.5.1 u32

Questo classificatore è tanto semplice quando potente: consente di esprimere condizioni sul valore di uno o più campi degli header IP,TCP,UDP,ICMP.

4.5.2 Fw (firewall IPTables)

Uno dei flag utilizzati nelle ACL (access list) del firewall linux consente di marcare con uno specifico valore i pacchetti che lo soddisfano.

Questo comando, presente nel file di configurazione del firewall (`/etc/sysconfig/iptables`) dice di marcare con il valore di 1 i pacchetti TCP appartenenti alla porta `80`, entranti dall'interfaccia eth1 e uscenti dall'interfaccia `eth0`.

```
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 80 -j MARK --set-mark 1
```

Il classificatore fw, quindi, è in realtà un classificatore piuttosto "stupido" che si limita a leggere il valore dell'etichetta posta dal firewall. In questo caso, quindi, l'intelligenza sta tutta nel firewall.

4.6 Introduzione alla CDL

I due principali tool di configurazione sono "`ip`" e "`tc`", sebbene, come abbiamo visto sopra, sia in realtà possibile usare applicazioni esterne, come il firewall IPTables per marcare (classificare) il traffico.

Mentre "`ip`" lo si può vedere come sostituto di "`ifconfig`" e "`route`", "`tc`" è invece utilizzato per la configurazione dello scheduler vero e proprio.

I comandi principali di cui abbiamo bisogno sono quelli che consentono di:

- Definire lo scheduler da usare;

- Creare classi;
- Definire filtri per l'associazione traffico->classe.

Digitando il comando senza alcun parametro, o un comando incompleto seguito dalla parola "`help`", è possibile ottenere un aiuto sulla sintassi rispettivamente generale e contestuale, come mostrato dai due esempi che seguono:

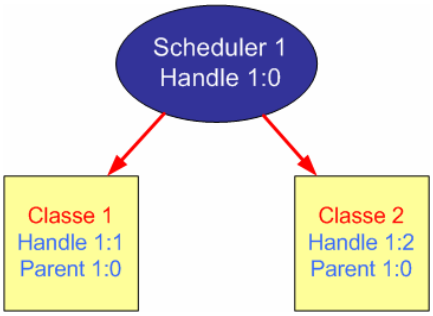
```
router# tc

Usage: tc [ OPTION ] OBJECT { COMMAND | help }
where  OBJECT := { qdisc | class | filter }
       OPTIONS := { -s[tatistics] | -d[etails] | - r[aw] }


router# tc qdisc add dev eth0 cbq help

Usage: … cbq bandwidth BPS avpkt BYTES [ mpu BYTES ]
            [ cell BYTES ] [ ewma LOG]
```

Ogni elemento (classificatore, scheduler,…) richiede alcuni parametri specifici che non avrebbero senso ad esempio per un altro elemento (es. isolated è usato solo da CBQ), mentre ce ne sono alcuni che sono necessari ed obbligatori per tutti:

- *Device*: indica l'interfaccia a cui si riferisce. Ogni interfaccia può infatti avere il suo scheduler;
- *Handle*: È un numero a due cifre, rappresentato nella forma X:Y, dove in genere i valori X:0 sono assegnati ai nodi dell'albero (quelli con figli, come scheduler 1 e 2) mentre X:Y sono utilizzati dalle foglie. Tutti i figli X:0 devono avere un handle della forma X:Y, ovvero ereditano X dal padre;
- *Parent*: Dovendo agganciare un elemento ad un altro, come un filtro ad un altro, come un filtro ad uno scheduler, occorre definire il nodo

di aggancio, il parent appunto;

- *Azione*: Con ogni comando si aggiunge o rimuove un elemento (Add, Remove) o si stampano a schermo alcune statistiche su uno già esistente (Show).

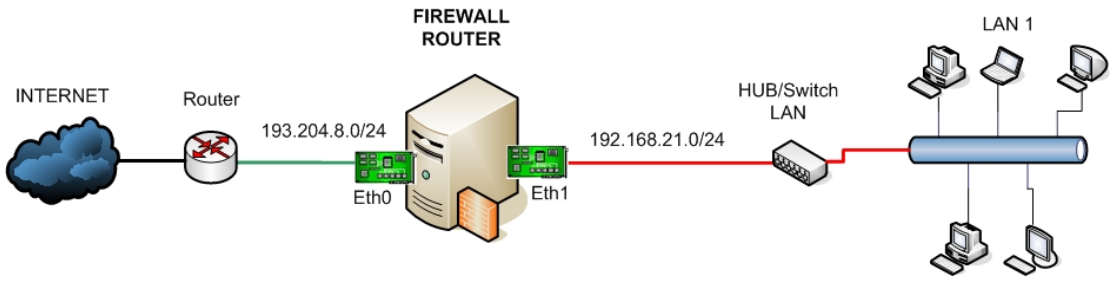


Se volessimo creare una terza classe sotto scheduler 1, dovremmo usare un comando con i seguenti parametri [26].

```
Device=eth0 Handle=1:3 Parent=1:0 Azione=add
```

4.7 Esempi pratici

Nella seguente immagine è visibile lo schema di rete in cui ho applicato il traffic shaping. Sono presenti due schede di rete a 10 Mbit una affacciata su Internet l'altra sulla LAN.



4.7.1 Prioritizzazione

Supponiamo che al traffico web e ftp deve essere applicata la priorità massima, quindi marchiamo questo tipo di traffico con il firewall attraverso la tabella MANGLE:

```
# Marca traffico HTTP

-A FORWARD -p TCP  -i eth1 -o eth0 --dport 80 -j MARK --set-mark 1
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 80 -j MARK --set-mark 1

# Marca traffico HTTPS

-A FORWARD -p TCP  -i eth1 -o eth0 --dport 443 -j MARK --set-mark 1
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 443 -j MARK --set-mark 1

# Marca traffico FTP

-A FORWARD -p TCP  -i eth1 -o eth0 --dport ftp  -j MARK --set-mark 1
-A FORWARD -s 0/0 -p tcp  ! --syn --sport ftp -j MARK --set-mark 1
-A FORWARD -s 192.168.0.0/24 -p tcp  --sport 1024: -d 0/0 --dport 1024:
-j MARK --set-mark 1
-A FORWARD -s 0/0 -p tcp ! --syn --sport 1024: -d 192.168.21.0/24  --
dport 1024: -j MARK --set-mark 1
```

Agganciamo lo scheduler con priorità all'interfaccia `eth0`:

```
router# tc  qdisc add dev eth0 root handle 1: prio
```

Mappiamo il traffico marcato dal firewall sulla classe a maggior priorità:

```
router# tc  filter add dev eth0 protocol ip parent 1:0 prio 1 handle 1
fw classied 1:1
```

4.7.2 Suddivisione di banda

Successivamente ho provato a dividere la banda nel seguente modo:

10% della banda (1Mbit) assegnato al traffico web (HTTP,HTTPS);

30% della banda (3Mbit) assegnato al traffico ftp;

60% della banda (6Mbit) assegnato al traffico rimanente.

Quando una delle tre classi di traffico non utilizza tutta la banda allocatagli, la parte inutilizzata deve poter essere riciclabile dalle altre. Può quindi capitare che in un dato istante, non essendoci né traffico ftp né web, tutti i 10Mbit a disposizione sono utilizzabili per il resto del traffico.

Marchiamo le tre categorie di traffico (web, ftp, altro) con tre diversi valori (1,2,3).

```
# Marca traffico HTTP
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 80 -j MARK --set-mark 1
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 80 -j MARK --set-mark 1

# Marca traffico HTTPS
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 443 -j MARK --set-mark 1
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 443 -j MARK --set-mark 1

# Marca traffico FTP
-A FORWARD -p TCP  -i eth1 -o eth0 --dport ftp  -j MARK --set-mark 1
-A FORWARD -s 0/0 -p tcp  ! --syn --sport ftp -j MARK --set-mark 1
-A FORWARD -s 192.168.0.0/24 -p tcp  --sport 1024: -d 0/0 --dport 1024:
-j MARK --set-mark 1
-A FORWARD -s 0/0 -p tcp ! --syn --sport 1024: -d 192.168.21.0/24  --
dport 1024: -j MARK --set-mark 1

# Marca traffico gtk-gnutella
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 6346:6348 -j MARK --set-mark
2
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 6346:6348 -j MARK --set-mark
2
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 2500 -j MARK --set-mark 2
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 2500 -j MARK --set-mark 2

# Marca traffico eMule
-A FORWARD -p TCP  -i eth0 -o eth1 --dport 4662 -j MARK --set-mark 3
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 4662 -j MARK --set-mark 3
-A FORWARD -p UDP  -i eth0 -o eth1 --dport 4672 -j MARK --set-mark 3
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 4672 -j MARK --set-mark 3
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 4661 -j MARK --set-mark 3
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 4665 -j MARK --set-mark 3
-A FORWARD -p UDP  -i eth0 -o eth1 --dport 4665 -j MARK --set-mark 3
-A FORWARD -p TCP  -i eth0 -o eth1 --dport 4711 -j MARK --set-mark 3
```

Agganciamo lo scheduler CBQ all'interfaccia eth0:

```
router# tc  qdisc add dev eth0 root handle 1: cbq bandwidth 10Mbit avpkt
        1000
```

Creiamo la classe radice:

```
router# tc  class add dev eth0 parent 1:10 classid 1:10 cbq bandwidth
        10Mbit rate 10Mbit allot 1514 weight 1Mbit prio 8 maxburst 20
        avpkt 1000
```

Creiamo le tre sottoclassi:

```
router# tc  class add dev eth0 parent 1:10 classid 1:11 cbq bandwidth
        10Mbit rate 1Mbit allot 1514 weight 100Kbit prio 5 maxburst 20
        avpkt 1000

router# tc  class add dev eth0 parent 1:10 classid 1:12 cbq bandwidth
        10Mbit rate 3Mbit allot 1514 weight 300Kbit prio 5 maxburst 20
        avpkt 1000

router# tc  class add dev eth0 parent 1:10 classid 1:13 cbq bandwidth
        10Mbit rate 6Mbit allot 1514 weight 600Kbit prio 5 maxburst 20
        avpkt 1000
```

Definiamo i filtri che associano il traffico alle classi:

```
router# tc  filter add dev eth0 protocol ip parent 1:0 prio 1 handle 1
        fw classid 1:11

router# tc  filter add dev eth0 protocol ip parent 1:0 prio 1 handle 2
        fw classid 1:12

router# tc  filter add dev eth0 protocol ip parent 1:0 prio 1 handle 3
        fw classid 1:13
```

A questo punto lo scheduler è configurato, ma per testarlo occorre vedere se i filtri si comportano bene assegnando il traffico alle giuste classi.

Con l'opzione `show` è possibile vedere i dettagli su classi, scheduler e filtri. In particolare si può avere un resoconto su quanti pacchetti e quanti byte le sono stati associati dai filtri, quanti sono stati scartati ecc.

Nel caso di CBQ viene mostrata anche la quantità di banda di cui una classe è andata in prestito. Utilizzando i flag "`-s`" e "`-details`" è possibile ottenere maggiori dettagli.

Successivamente vengono visualizzate le statistiche, dopo che ho impostato la banda riservata al traffico web a 100Kbit e ho generato parecchio traffico web con l'applicazione Teleport Pro.

```
router# tc –s qdisc show dev eth0

qdisc cbq 1: rate 10Mbit (bounded,isolated) prio no-transmit
  Sent 231000237 bytes 295680 pkts (dropped 0, overlimits 0)
   borrowed 0 overactions 0 avgidle 569 undertime 0


router# tc –s class show dev eth0

class cbq 1:11 parent 1:10 rate 10Kbit prio 5
 Sent 1002461 bytes 16613 pkts (dropped 0, overlimits 0)
  borrowed 8375 overactions 0 avgidle -485152 undertime 500643
class cbq 1: root rate 10Mbit (bounded,isolated) prio no-transmit
 Sent 227203821 bytes 289749 pkts (dropped 0, overlimits 0)
  borrowed 0 overactions 0 avgidle 587 undertime 0
class cbq 1:10 parent 1: rate 10Mbit prio no-transmit
 Sent 153601915 bytes 197964 pkts (dropped 0, overlimits 0)
  borrowed 8375 overactions 0 avgidle 605 undertime 0
class cbq 1:13 parent 1:10 rate 6Mbit prio 5
 Sent 152590378 bytes 181240 pkts (dropped 0, overlimits 0)
  borrowed 0 overactions 0 avgidle 11825 undertime 0
class cbq 1:12 parent 1:10 rate 3Mbit prio 5
 Sent 9076 bytes 111 pkts (dropped 0, overlimits 0)
  borrowed 0 overactions 0 avgidle 41391 undertime 0
```

Come si nota dal codice sovrastante la 1:11 è andata in prestito di banda (borrowed!=0). Se poi si aggiungono i due flag bounded e isolated, rimanendo in condizioni di traffico in eccesso per la classe 1:11, possiamo vedere che il surplus viene scartato (dropped!=0 e borrowed=0):

```
router# tc –s qdisc show dev eth0

qdisc cbq 1: rate 10Mbit (bounded,isolated) prio no-transmit
 Sent 278653153 bytes 355342 pkts (dropped 62, overlimits 51045)
 backlog 99p
  borrowed 0 overactions 0 avgidle 587 undertime 0
```

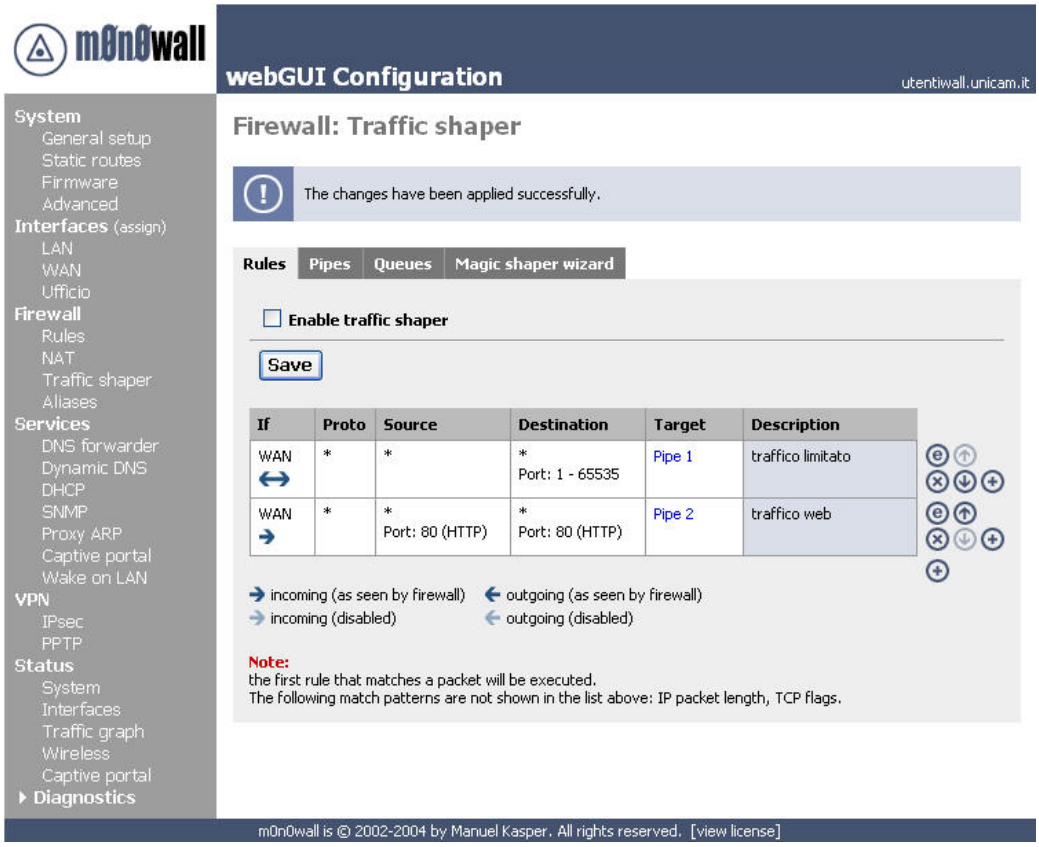
```
router# tc –s class show dev eth0

class cbq 1:11 parent 1:10 rate 10Kbit (bounded,isolated) prio 5
 Sent 1702242 bytes 29239 pkts (dropped 76, overlimits 140964)
 backlog 100p
  borrowed 0 overactions 8090 avgidle -15307 undertime 22421
class cbq 1: root rate 10Mbit (bounded,isolated) prio no-transmit
 Sent 280531062 bytes 348840 pkts (dropped 0, overlimits 0)
  borrowed 0 overactions 0 avgidle 624 undertime 0
class cbq 1:10 parent 1: rate 10Mbit prio no-transmit
 Sent 198182590 bytes 247619 pkts (dropped 0, overlimits 0)
  borrowed 11132 overactions 0 avgidle 624 undertime 0
class cbq 1:13 parent 1:10 rate 6Mbit prio 5
 Sent 196989331 bytes 227629 pkts (dropped 0, overlimits 0)
  borrowed 0 overactions 0 avgidle 11825 undertime 0
class cbq 1:12 parent 1:10 rate 3Mbit prio 5
 Sent 13624 bytes 164 pkts (dropped 0, overlimits 0)
  borrowed 0 overactions 0 avgidle 41391 undertime 0
```

È interessante osservare che 76 pacchetti destinati alla classe 1:11 sono stati scartati e la coda (backlog) associata contiene 100 elementi (il valore massimo), ossia è satura. Questo è un chiaro segnale che la banda assegnata a tale classe non è sufficiente.

Per quanto flessibile ci sono anche alcune limitazioni, ad esempio non è possibile configurare uno scheduler da applicare a tutto il traffico in uscita o a tutto quello in ingresso; occorre sempre agganciarsi ad un’interfaccia specifica.

4.8 Implementazione del Traffic Shaping con M0n0wall

È possibile implementare il Traffic Shaping anche con M0n0wall, tramite le apposite form, visibili nelle immagini successive.

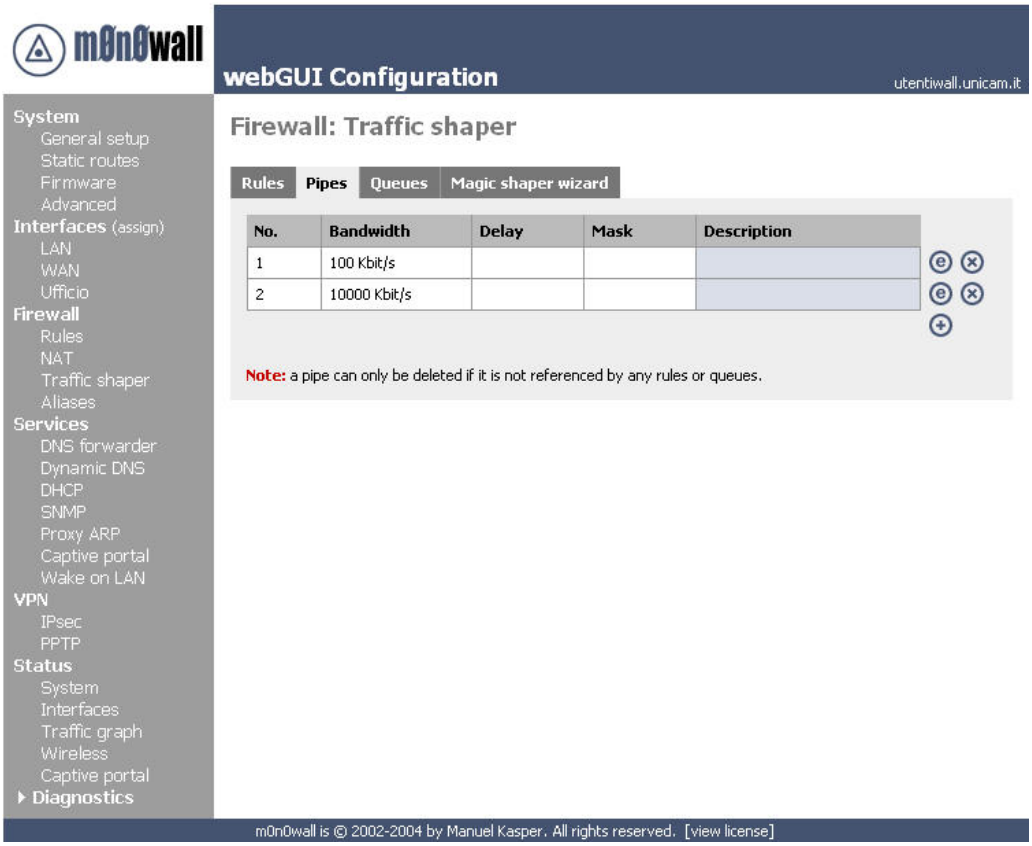


Nell'immagine è possibile vedere l'impostazione di due regole. La prima regola controlla il traffico, entrante ed uscente, dall'interfaccia connessa ad internet (WAN), per qualsiasi protocollo, per qualsiasi indirizzo IP sorgente e destinazione e per il range di porte che va da 1 a 65535. Tale regola è legata alla *pipe 1* che come vedremo nell'immagine successiva non fa altro che impostare il valore per definire la quantità di banda riservata a tale traffico.

La seconda regola invece, definisce, tramite la *pipe 2*, la banda riservata al traffico che entra dall'interfaccia connessa ad internet (WAN), per qualsiasi

protocollo, per qualsiasi indirizzo IP sorgente e destinazione e per la porta 80 (HTTP).

L'immagine successiva mostra come vengono realizzate le *pipes* che vengono assegnante alle regole, visibili nell'immagine sovrastante.

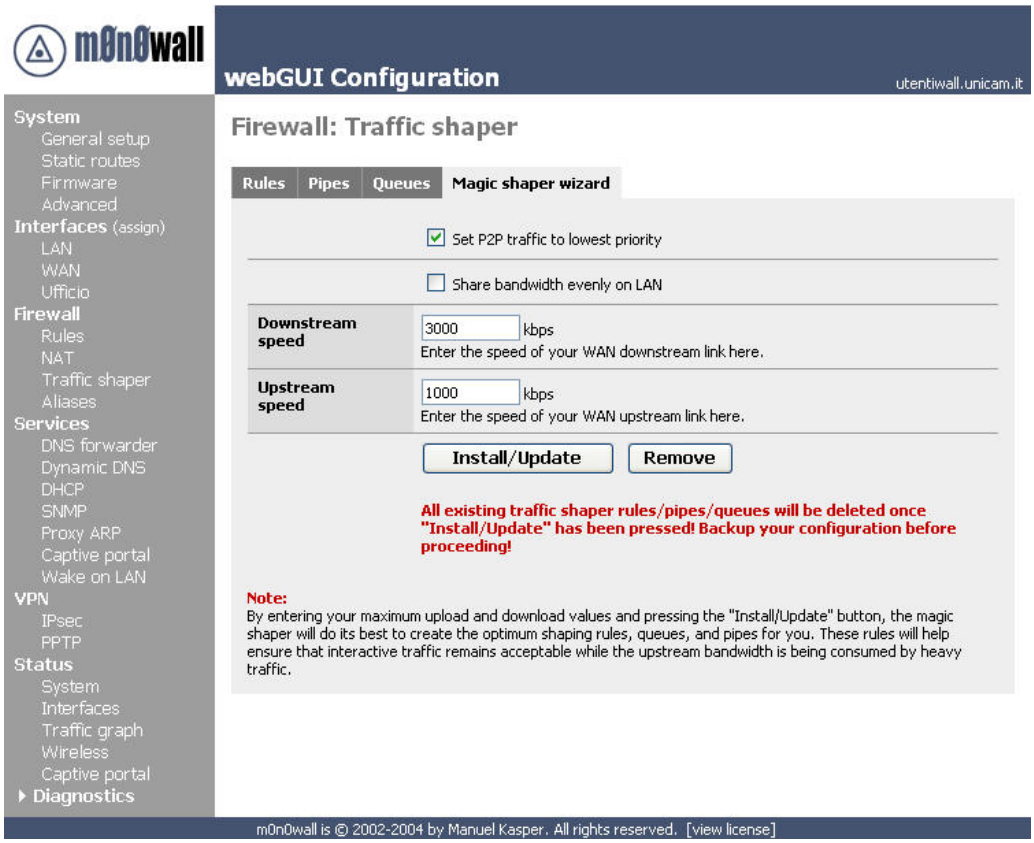


L'immagine sovrastante mostra come sono state definite le due *pipes* utilizzate.

Alla prima pipe è stata assegnata una banda di 100 Kbit/s mentre alla seconda 10000 Kbit/s.

Inoltre, nel caso in cui nella rete LAN sono presenti dei client p2p, è possibile definire la quantità di banda da assegnare, in ingesso e uscita dalla LAN. È possibile anche assegnare al traffico p2p una priorità più bassa, rispetto all'altro traffico passante attraverso il firewall. L'immagine successiva ci mostra la form per

impostare tali valori.

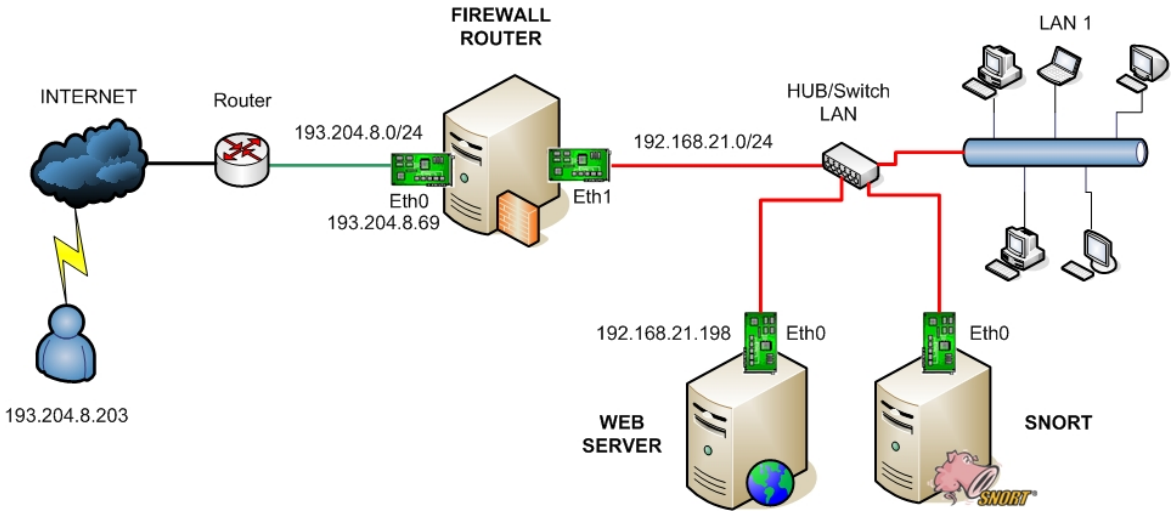


CAPITOLO 5

UTILIZZO DI EXPLOIT PER TESTARE IL SISTEMA CREATO

5.0 Web Server pubblico all'interno della LAN

Lo schema di rete sottostante mostra una rete con incluso un Web Server all'interno della rete LAN.



5.0.1 Implementazione con IPTables

È possibile implementare, con IPTables, un Web server pubblico, presente all'interno della rete LAN, aggiungendo la seguente regola sulla tabella NAT:

```
 -A PREROUTING -d 193.204.8.69 -p tcp --dport 80 -j DNAT --to-destination
192.168.21.198
```

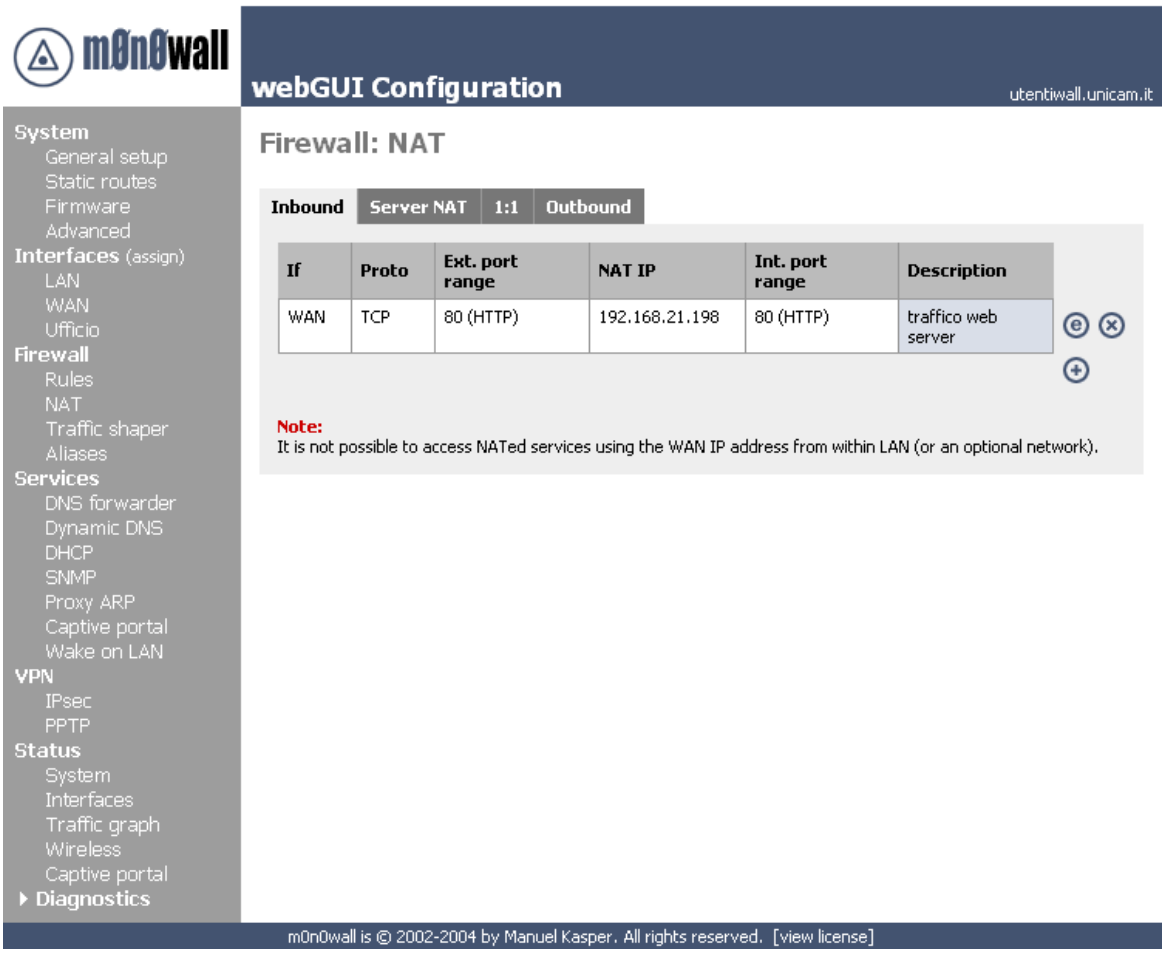
In questo modo si comunica al kernel della macchina firewall (`193.204.8.69`) di reindirizzare tutti i pacchetti destinati all'indirizzo pubblico

`193.204.8.69` (opzione `-d` destination) e alla porta TCP (`-p tcp`) 80 (`--dport 80`) all'indirizzo IP `192.168.21.198` (opzione `--to-destination`). La porta di destinazione è sottintesa (`80`).

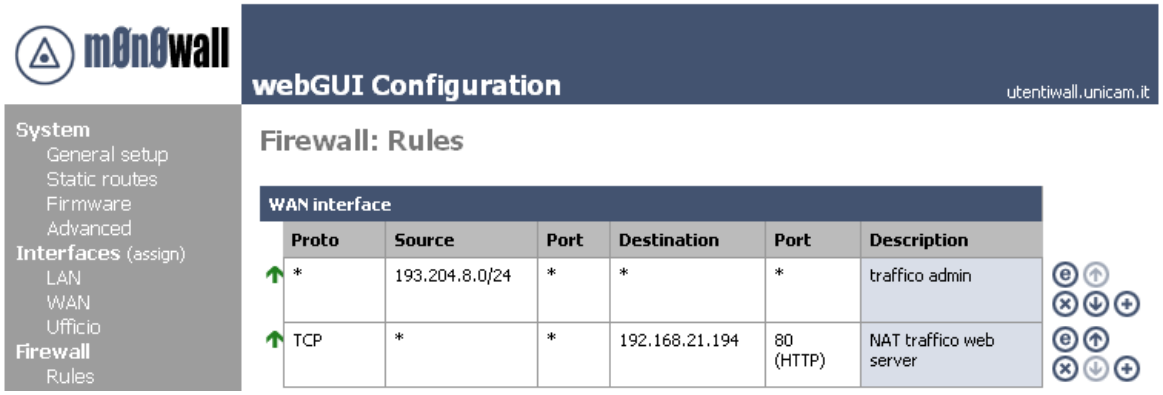
In questo modo qualsiasi utente di INTERNET (ad es: `193.204.8.203`) può visualizzare, tramite browser, la pagina web contenuta sulla macchina con indirizzo privato `192.168.21.198`, digitando semplicemente l'indirizzo pubblico `193.204.8.69`.

5.0.2 Implementazione con M0n0wall

È possibile implementare lo stesso tipo di configurazione, anche cono M0n0wall, come risulta evidente nell'immagine successiva.



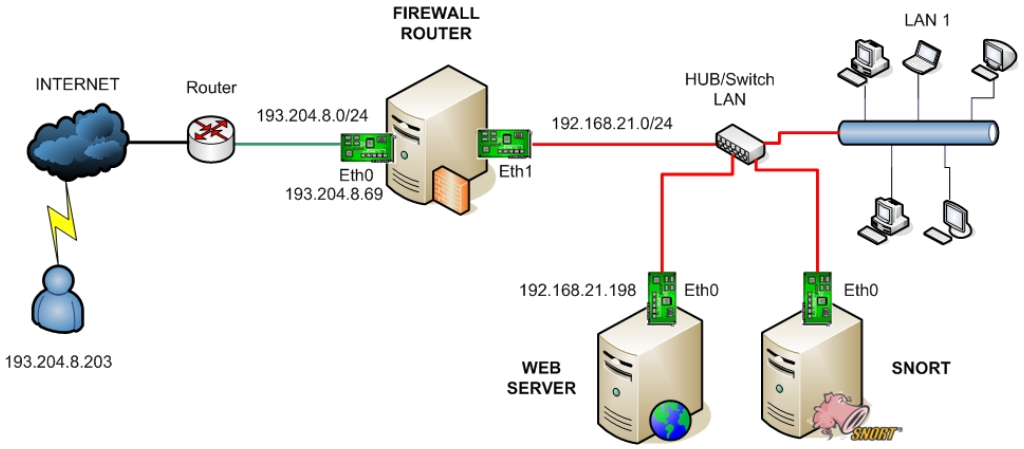
Impostando comunque la regola visibile nell'immagine sovrastante, il sistema aggiunge automaticamente un ulteriore regola, come è evidente nell'immagine successiva.



Anche in questo caso qualsiasi utente di internet (ad es: `193.204.8.203`) può visualizzare, tramite browser, la pagina web contenuta sulla macchina con indirizzo privato `192.168.21.198`, digitando semplicemente l'indirizzo pubblico `193.204.8.69`.

5.1 Problemi di sicurezza

Nonostante l'adozione di firewall e IDS, la rete potrebbe essere ancora soggetta ad attacchi; infatti se è presente un web server all'interno della rete LAN visibile all'esterno (internet), questo potrebbe essere preso di mira dai malintenzionati.



Lo schema sovrastate mostra la rete LAN con incluso un Web Server pubblico, quindi qualsiasi utente di internet potrebbe tranquillamente usufruire del servizio messo a disposizione e addirittura attaccare il Web server.

Comunque gli attacchi per questa rete possono provenire, non solo da internet ma anche:

- Sede remota;
- Client remoto legittimo;
- Client remoto illegittimo;
- Client locale legittimo;
- Client locale illegittimo.

Nel caso in cui poi siano stati compromessi dei sistemi, è possibile che gli attacchi provengano da qualunque punto compresi, server, router e firewall stessi.

5.2 I limiti di un firewall

Vi sono due tipologie principali che generano problemi di sicurezza:

- *Configurazione non corretta*: quando tramite una configurazione apposita del "sistema firewall" si sarebbero potuti evitare degli incidenti o quando si riscontrano impostazioni che forniscono dei punti di attacco superflui rispetto al funzionamento del sistema stesso;

- *Problematiche tecniche*: quando nonostante siano state impostate le configurazioni a regola d'arte sul firewall e non vi siano punti di attacco superflui, sia possibile sfruttando degli errori di programmazione o delle vulnerabilità del firewall stesso forzare le regole imposte ed accedere indebitamente ad un componente che "teoricamente" si riteneva protetto [29].

5.3 I limiti di un'IDS

Anche gli IDS purtroppo hanno dei limiti, infatti oltre a generare falsi positivi e negativi, sono basati su regole che come gli antivirus devono essere costantemente aggiornate e nonostante questo, sono in grado di individuare soltanto attacchi conosciuti e in chiaro; infatti se gli attacchi vengono eseguiti in maniera criptata, l'IDS non è in grado di individuarli.

5.4 Due attacchi in dettaglio

Per mostrare le problematiche precedentemente descritte ho testato il sistema, firewall/IDS da me configurato, con due exploit per il Web server Apache.

Il primo script *"Remotely Exploitable Overflow In mod_mylo For Apache"* [30] è un exploit in grado di concedere ad un malintenzionato, da remoto, la possibilità di sovrascrivere i dati salvati nello stack e consentirgli di eseguire comandi arbitrari con i privilegi del demone Apache. Il sistema su cui ho provato tale script è Linux Red Hat 7.3 con il Web server Apache 1.3.23 e mod_mylo versione 0.2.1.

Per poter compilare il sorgente dello script ho utilizzato il seguente comando:

```
193.204.8.203 #gcc –o exp1 exp1.c
```

Successivamente ho lanciato lo script, nel seguente modo:

```
193.204.8.203 #./exp1 -h


Apache + mod_mylo remote exploit
By Carl Livitt (carllivitt at hush dot com)

Arguments:
 -t target Attack 'target' host
 -T platform Use parameters for target 'platform'
 -h This help.

Available platforms:
 0. SuSE 8.1, Apache 1.3.27 (installed from source) (default)
 1. RedHat 7.2, Apache 1.3.20 (installed from RPM)
 2. RedHat 7.3, Apache 1.3.23 (installed from RPM)
 3. FreeBSD 4.8, Apache 1.3.27 (from Ports)
```

Il codice sovrastante mostra la sintassi da usate per poter eseguire lo script, mentre quello successivo mostra come l'ho eseguito.

```
193.204.8.203 #./exp1 –t 193.204.8.69 –T 2
```

A questo punto, al Web Server presente sulla macchina con indirizzo IP `192.168.21.198` arrivano, sulla porta `80` dei pacchetti che provano a forzare l'esecuzione del codice arbitrario con i privilegi di Apache.

A questo punto possiamo dire che il firewall fa passare tranquillamente tale traffico perché è destinato alla porta `80`, ma l'IDS è in grado di rilevarlo, come risulta evidente dal file di log di snort (`/var/log/snort/alert`).

```
03/03-13:15:26.597136  [**] [1:648:7] SHELLCODE x86 NOOP [**]
[Classification: Executable code was detected] [Priority: 1] {TCP}
193.204.8.203:1632 -> 192.168.21.198:80

03/03-13:15:26.597136  [**] [119:4:1] (http_inspect) BARE BYTE UNICODE
ENCODING [**] [Classification: Executable code was detected] [Priority:
1] {TCP} 193.204.8.203:1632 -> 192.168.21.198:80
```

A questo punto interviene Guardian, descritto precedentemente, che provvede ad impostare una regola sul firewall, per bloccare qualsiasi traffico

proveniente dall'indirizzo IP (193.204.8.203) del malintenzionato, come risulta evidente dal codice sottostante.

```
193.204.8.69 #/sbin/iptables -L

Chain INPUT (policy DROP)
target     prot opt source               destination
DROP       all  --  193.204.8.203        anywhere

Chain FORWARD (policy DROP)
target     prot opt source               destination
DROP       all  --  193.204.8.203        192.168.21.0/24
DROP       all  --  193.204.8.203        193.205.95.0/24
```

Il secondo script "*Apache Multiple Space Header DoS (Multi-Threaded Exploit)*" [31] è un attacco Denial of Service in grado di inviare da remoto delle richieste http malformate da parte di un malintenzionato. Ciò non fa altro che bloccare il servizio del Web Server. Ho provato l'efficacia di tale script su una Red Hat 9.0 con Apache 2.0.40.

Prima di tutto ho compilato il sorgente dello script utilizzando il seguente comando:

```
193.204.8.203 # gcc -lpthread -o exp2 exp2.c
```

Successivamente ho lanciato lo script, nel seguente modo:

```
193.204.8.203 #./exp2 193.204.8.69
```

A questo punto, al Web Server presente sulla macchina con indirizzo IP 192.168.21.198 arrivano, sulla porta 80 delle richieste http malformate che vanno a saturare le risorse e quindi a bloccare il servizio.

Il firewall fa passare tranquillamente tale traffico perché è destinato alla porta 80. Anche l'IDS comunque non è in grado di rilevare questo tipo di attacco, come ho potuto osservare dal file di log (/var/log/snort/alert).

CONCLUSIONI

Come abbiamo visto con l'ultimo capitolo, nonostante l'impiego del firewall e di un'IDS, il sistema non garantisce comunque una completa sicurezza.

Infatti non è sufficiente avere un firewall e un IDS per risolvere, tutti i problemi di sicurezza, e né si può pensare che installare un firewall renda inutili i backup di sicurezza dei dati, che conviene sempre programmare opportunamente, indipendentemente da quanto sia ristretta la politica di sicurezza adottata, perché rappresentano l'unico modo di tutelarsi da guasti o distruzione dell'hardware.

C'è da dire quindi che per mantenere un alto livello di sicurezza, l'amministratore deve costantemente applicare le patch agli applicativi e ai sistemi operativi, impiegare un sistema antivirus costantemente aggiornato, per rilevare eventuali virus, e controllare costantemente i file di log alla ricerca di eventuali intrusioni non bloccate dal firewall e non rilevate dall'IDS.

APPENDICE

Ho riportato in questa appendice gli script di configurazione di IPTables e M0n0wall da me realizzati.

IPTables (/etc/sysconfig/iptables)

```
# Generated by iptables-save v1.2.7a on Thu Feb 26 15:21:16 2004
*nat
:PREROUTING ACCEPT [181:28365]
:POSTROUTING ACCEPT [7:444]
:OUTPUT ACCEPT [18:1128]
#-A POSTROUTING  -o eth0 -s 192.168.21.0/24  -j MASQUERADE
-A POSTROUTING -s 192.168.21.0/24 -o eth0 -j SNAT --to 193.204.8.132-
193.204.8.134
COMMIT
# Completed on Thu Feb 26 15:21:16 2004
# Generated by iptables-save v1.2.7a on Thu Feb 26 15:21:16 2004
*mangle
:PREROUTING ACCEPT [7722:680159]
:INPUT ACCEPT [7714:679679]
:FORWARD ACCEPT [8:480]
:OUTPUT ACCEPT [6307:494203]
:POSTROUTING ACCEPT [6315:494683]
# Marca traffico HTTP
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 80 -j MARK --set-mark 1
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 80 -j MARK --set-mark 1
# Marca traffico HTTPS
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 443 -j MARK --set-mark 1
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 443 -j MARK --set-mark 1
# Marca traffico FTP
-A FORWARD -p TCP  -i eth1 -o eth0 --dport ftp  -j MARK --set-mark 1
-A FORWARD -s 0/0 -p tcp  ! --syn --sport ftp -j MARK --set-mark 1
-A FORWARD -s 192.168.0.0/24 -p tcp  --sport 1024: -d 0/0 --dport 1024:
-j MARK --set-mark 1
-A FORWARD -s 0/0 -p tcp ! --syn --sport 1024: -d 192.168.21.0/24  --
dport 1024: -j MARK --set-mark 1
# Marca traffico gtk-gnutella
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 6346:6348 -j MARK --set-mark
2
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 6346:6348 -j MARK --set-mark
2
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 2500 -j MARK --set-mark 2
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 2500 -j MARK --set-mark 2
```

```
# Marca traffico eMule
-A FORWARD -p TCP  -i eth0 -o eth1 --dport 4662 -j MARK --set-mark 3
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 4662 -j MARK --set-mark 3
-A FORWARD -p UDP  -i eth0 -o eth1 --dport 4672 -j MARK --set-mark 3
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 4672 -j MARK --set-mark 3
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 4661 -j MARK --set-mark 3
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 4665 -j MARK --set-mark 3
-A FORWARD -p UDP  -i eth0 -o eth1 --dport 4665 -j MARK --set-mark 3
-A FORWARD -p TCP  -i eth0 -o eth1 --dport 4711 -j MARK --set-mark 3
COMMIT
# Completed on Thu Feb 26 15:21:16 2004
# Generated by iptables-save v1.2.7a on Thu Feb 26 15:21:16 2004
*filter
:FORWARD DROP [0:0]
:INPUT DROP [0:0]
:OUTPUT DROP [0:0]

-A OUTPUT -p icmp -j ACCEPT
-A OUTPUT -p ALL -o lo -s 127.0.0.1 -j ACCEPT
-A OUTPUT -p TCP -o eth0 --dport 1863 -j ACCEPT
-A OUTPUT -p UDP -o eth0 --dport 1863 -j ACCEPT
-A OUTPUT -p TCP -o eth0 --dport 5190 -j ACCEPT
-A OUTPUT -p UDP -o eth0 --dport 5190 -j ACCEPT
-A OUTPUT -p TCP -o eth0 --dport 53 -j ACCEPT
-A OUTPUT -p UDP -o eth0 --dport 53 -j ACCEPT
-A OUTPUT -p TCP -o eth0 --dport 443 -j ACCEPT
-A OUTPUT -p TCP -o eth0 --dport 80 -j ACCEPT
-A OUTPUT -p TCP -o eth0 --dport ftp  -j ACCEPT
-A OUTPUT -s 0/0 -p tcp  ! --syn --sport ftp -j ACCEPT
-A OUTPUT -s 193.204.8.0/24 -p tcp  --sport 1024: -d 0/0 --dport 1024: -
j ACCEPT
-A OUTPUT -s 0/0  -p tcp ! --syn --sport 1024: -d 193.204.8.0/24  --
dport 1024: -j ACCEPT
-A OUTPUT -p TCP -m state --state RELATED,ESTABLISHED -j ACCEPT
-A OUTPUT -m limit --limit 50/minute --limit-burst 3 -j LOG --log-level
debug

-A INPUT -p icmp -j ACCEPT
-A INPUT -p ALL -i lo -d 127.0.0.1 -j ACCEPT
-A INPUT -p TCP -i eth1 -d 192.168.21.1  --dport 22 -j ACCEPT
-A INPUT -p TCP -s 193.205.95.49 -d 193.204.8.69 --dport 22 -j ACCEPT
-A INPUT -p TCP -s 192.168.21.200 -d 193.204.8.69 --dport 22 -j ACCEPT
-A INPUT -p TCP -i eth1  --dport 80 -j ACCEPT
-A INPUT -i eth0 -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -p udp -m udp ! --sport 137:139 -j LOG --log-level debug
-A INPUT -m limit --limit 50/minute --limit-burst 3 -j LOG --log-level
debug

# Traffico interfaccia eth1 rete 192.168.21.0

# Traffico gtk-gnutella
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 6346:6348 -j ACCEPT
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 6346:6348 -j ACCEPT
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 2500 -j ACCEPT
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 2500 -j ACCEPT

# Traffico eMule
-A FORWARD -p TCP  -i eth0 -o eth1 --dport 4662 -j ACCEPT
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 4662 -j ACCEPT
```

```
-A FORWARD -p UDP  -i eth0 -o eth1 --dport 4672 -j ACCEPT
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 4672 -j ACCEPT
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 4661 -j ACCEPT
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 4665 -j ACCEPT
-A FORWARD -p UDP  -i eth0 -o eth1 --dport 4665 -j ACCEPT
-A FORWARD -p TCP  -i eth0 -o eth1 --dport 4711 -j ACCEPT

-A FORWARD -p TCP  -i eth1 -o eth0 --dport 53 -j ACCEPT
-A FORWARD -p UDP  -i eth1 -o eth0 --dport 53 -j ACCEPT
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 80 -j ACCEPT
-A FORWARD -p TCP  -i eth1 -o eth0 --dport 443 -j ACCEPT
-A FORWARD -p TCP  -i eth1 -o eth0 --dport ftp -j ACCEPT
-A FORWARD -s 0/0 -p tcp  ! --syn --sport ftp -j ACCEPT
-A FORWARD -s 192.168.21.0/24 -p tcp  --sport 1024: -d 0/0 --dport 1024:
-j ACCEPT
-A FORWARD -s 0/0 -p tcp ! --syn --sport 1024: -d 192.168.21.0/24  --
dport 1024: -j ACCEPT
-A FORWARD -p icmp -i eth1 -o eth0 -j ACCEPT
-A FORWARD -i eth0 -o eth1 -m state --state ESTABLISHED,RELATED -j
ACCEPT
-A FORWARD -m limit --limit 50/minute --limit-burst 3 -j LOG --log-level
debug

#Traffico interfaccia eth2 rete 193.205.95.0

# Traffico eMule
-A FORWARD -p TCP  -i eth0 -o eth2 --dport 4662 -j ACCEPT
-A FORWARD -p TCP  -i eth2 -o eth0 --dport 4662 -j ACCEPT
-A FORWARD -p UDP  -i eth0 -o eth2 --dport 4672 -j ACCEPT
-A FORWARD -p UDP  -i eth2 -o eth0 --dport 4672 -j ACCEPT
-A FORWARD -p TCP  -i eth2 -o eth0 --dport 4661 -j ACCEPT
-A FORWARD -p UDP  -i eth2 -o eth0 --dport 4665 -j ACCEPT
-A FORWARD -p UDP  -i eth0 -o eth2 --dport 4665 -j ACCEPT
-A FORWARD -p TCP  -i eth0 -o eth2 --dport 4711 -j ACCEPT

# Traffico verso il web nella rete privata

-A FORWARD -p TCP  -i eth0 -o eth2 -d 193.205.95.0/24 --dport 53 -j
ACCEPT
-A FORWARD -p UDP  -i eth0 -o eth2 -d 193.205.95.0/24 --dport 53 -j
ACCEPT
-A FORWARD -p TCP  -i eth0 -o eth2 -d 193.205.95.0/24 --dport 80 -j
ACCEPT
-A FORWARD -p TCP  -i eth0 -o eth2 -d 193.205.95.0/24 --dport 443 -j
ACCEPT
-A FORWARD -p icmp -i eth0 -o eth2 -d 193.205.95.0/24 -j ACCEPT
-A FORWARD -i eth2 -o eth0 -m state --state ESTABLISHED,RELATED -j
ACCEPT
-A FORWARD -m limit --limit 50/minute --limit-burst 3 -j LOG --log-level
debug

# Traffico dalla rete privata verso internet

-A FORWARD -p TCP -i eth2 -o eth0 --dport 1863 -j ACCEPT
-A FORWARD -p UDP -i eth2 -o eth0 --dport 1863 -j ACCEPT
-A FORWARD -p TCP -i eth2 -o eth0 --dport 5190 -j ACCEPT
-A FORWARD -p UDP -i eth2 -o eth0 --dport 5190 -j ACCEPT

-A FORWARD -p TCP  -i eth2 -o eth0 --dport 23 -j ACCEPT
```

```
-A FORWARD -p TCP  -i eth2 -o eth0 --dport 53 -j ACCEPT
-A FORWARD -p UDP  -i eth2 -o eth0 --dport 53 -j ACCEPT
-A FORWARD -p TCP  -i eth2 -o eth0 --dport 80 -j ACCEPT
-A FORWARD -p TCP  -i eth2 -o eth0 --dport 443 -j ACCEPT
-A FORWARD -p TCP  -i eth2 -o eth0 --dport ftp -j ACCEPT
-A FORWARD -s 0/0 -p tcp  ! --syn --sport ftp -j ACCEPT
-A FORWARD -s 193.205.95.0/24 -p tcp  --sport 1024: -d 0/0 --dport 1024:
-j ACCEPT
-A FORWARD -s 0/0 -p tcp ! --syn --sport 1024: -d 193.205.95.0/24  --
dport 1024: -j ACCEPT
-A FORWARD -p icmp -i eth2 -o eth0 -j ACCEPT
-A FORWARD -p TCP  -i eth2 -o eth0 --dport 22 -j ACCEPT
-A FORWARD -i eth0 -o eth2 -m state --state ESTABLISHED,RELATED -j
ACCEPT
-A FORWARD -m limit --limit 50/minute --limit-burst 3 -j LOG --log-level
debug
COMMIT
# Completed on Thu Feb 26 15:21:16 2004
```

M0n0wall (/conf/config.xml)

```
<?xml version="1.0" ?>
  <m0n0wall>
  <version>1.4</version>
  <system>
  <hostname>m0n0wall</hostname>
  <domain>local</domain>
  <dnsallowoverride />
  <username>admin</username>
  <password>$1$ofCRML9K$YA27L1l1nwqYn1XAonVYK1</password>
  <timezone>Etc/UTC</timezone>
  <time-update-interval>300</time-update-interval>
  <timeservers>pool.ntp.org</timeservers>
  <webgui>
  <protocol>http</protocol>
  <port />
  <certificate />
  <private-key />
  </webgui>
  <disableconsolemenu />
  <dnsserver>193.204.8.13</dnsserver>
  <dnsserver>193.204.8.26</dnsserver>
  </system>
  <interfaces>
  <lan>
  <if>xl0</if>
  <ipaddr>192.168.21.1</ipaddr>
  <subnet>24</subnet>
  </lan>
  <wan>
  <if>rl0</if>
  <ipaddr>193.204.8.135</ipaddr>
  <subnet>24</subnet>
  <gateway>193.204.8.14</gateway>
  <spoofmac />
  <mtu />
  </wan>
  <opt1>
  <if>xl1</if>
  <descr>Ufficio</descr>
  <ipaddr>193.205.95.1</ipaddr>
  <subnet>24</subnet>
  <bridge />
  <enable />
  </opt1>
  </interfaces>
  <staticroutes />
  <pppoe />
  <pptp />
  <dyndns>
  <type>dyndns</type>
  <username />
  <password />
  <host />
  <mx />
  </dyndns>
  <dhcpd>
```

```
<lan>
<enable />
<range>
<from>192.168.21.100</from>
<to>192.168.21.200</to>
</range>
<defaultleasetime />
<maxleasetime />
</lan>
<opt1>
<range>
<from>193.205.95.100</from>
<to>193.205.95.200</to>
</range>
<defaultleasetime>7200</defaultleasetime>
<maxleasetime>86400</maxleasetime>
<enable />
</opt1>
</dhcpd>
<pptpd>
<mode />
<redir />
<localip />
<remoteip />
</pptpd>
<dnsmasq>
<enable />
<regdhcp />
</dnsmasq>
<snmpd>
<syslocation />
<syscontact />
<rocommunity>public</rocommunity>
</snmpd>
<diag>
<ipv6nat>
<ipaddr />
</ipv6nat>
</diag>
<bridge />
<syslog>
<reverse />
<nentries>50</nentries>
<remoteserver />
</syslog>
<nat>
<advancedoutbound>
<enable />
</advancedoutbound>
</nat>
<filter>
<rule>
<type>pass</type>
<interface>wan</interface>
<source>
<address>193.204.8.0/24</address>
</source>
<destination>
<any />
```

```
</destination>
<descr>traffico admin</descr>
</rule>
<rule>
<type>pass</type>
<interface>wan</interface>
<protocol>tcp/udp</protocol>
<source>
<any />
</source>
<destination>
<any />
</destination>
<log />
<descr>traffico ftp</descr>
</rule>
<rule>
<type>pass</type>
<interface>opt1</interface>
<protocol>icmp</protocol>
<source>
<network>opt1</network>
</source>
<destination>
<any />
</destination>
<descr>traffico icmp</descr>
</rule>
<rule>
<type>pass</type>
<interface>opt1</interface>
<protocol>tcp/udp</protocol>
<source>
<network>opt1</network>
</source>
<destination>
<any />
<port>8080</port>
</destination>
<log />
<descr>traffico proxy</descr>
</rule>
<rule>
<type>pass</type>
<interface>opt1</interface>
<protocol>tcp/udp</protocol>
<source>
<network>opt1</network>
</source>
<destination>
<any />
<port>80</port>
</destination>
<log />
<descr>traffico http</descr>
</rule>
<rule>
<type>pass</type>
<interface>opt1</interface>
```

```
<protocol>tcp/udp</protocol>
<source>
<network>opt1</network>
</source>
<destination>
<any />
<port>443</port>
</destination>
<log />
<descr>traffico https</descr>
</rule>
<rule>
<type>pass</type>
<interface>opt1</interface>
<protocol>tcp/udp</protocol>
<source>
<network>opt1</network>
</source>
<destination>
<any />
<port>53</port>
</destination>
<log />
<descr>traffico dns</descr>
</rule>
<rule>
<type>pass</type>
<interface>opt1</interface>
<protocol>tcp/udp</protocol>
<source>
<network>opt1</network>
</source>
<destination>
<any />
<port>22</port>
</destination>
<log />
<descr>traffico ssh</descr>
</rule>
<rule>
<type>pass</type>
<interface>opt1</interface>
<protocol>tcp/udp</protocol>
<source>
<network>opt1</network>
</source>
<destination>
<any />
<port>1863</port>
</destination>
<log />
<descr>traffico msn</descr>
</rule>
<rule>
<type>pass</type>
<interface>opt1</interface>
<protocol>tcp/udp</protocol>
<source>
<network>opt1</network>
```

```
</source>
<destination>
<any />
<port>5190</port>
</destination>
<log />
<descr>traffico icq</descr>
</rule>
<rule>
<type>pass</type>
<interface>opt1</interface>
<protocol>tcp/udp</protocol>
<source>
<network>opt1</network>
</source>
<destination>
<any />
<port>21</port>
</destination>
<log />
<frags />
<descr>traffico ftp</descr>
</rule>
<rule>
<type>pass</type>
<interface>opt1</interface>
<protocol>tcp/udp</protocol>
<source>
<any />
</source>
<destination>
<any />
<port>119</port>
</destination>
<descr>traffico news</descr>
</rule>
<rule>
<type>pass</type>
<interface>lan</interface>
<protocol>icmp</protocol>
<source>
<network>lan</network>
</source>
<destination>
<any />
</destination>
<descr>traffico icmp</descr>
</rule>
<rule>
<type>pass</type>
<interface>lan</interface>
<protocol>tcp</protocol>
<source>
<network>lan</network>
</source>
<destination>
<any />
<port>8080</port>
</destination>
```

```
<descr>traffico proxy</descr>
</rule>
<rule>
<type>pass</type>
<interface>lan</interface>
<protocol>tcp/udp</protocol>
<source>
<network>lan</network>
</source>
<destination>
<any />
<port>80</port>
</destination>
<log />
<descr>traffico http</descr>
</rule>
<rule>
<type>pass</type>
<interface>lan</interface>
<protocol>tcp/udp</protocol>
<source>
<network>lan</network>
</source>
<destination>
<any />
<port>443</port>
</destination>
<log />
<descr>traffico https</descr>
</rule>
<rule>
<type>pass</type>
<interface>lan</interface>
<protocol>tcp/udp</protocol>
<source>
<network>lan</network>
</source>
<destination>
<any />
<port>53</port>
</destination>
<log />
<descr>traffico dns</descr>
</rule>
<rule>
<type>pass</type>
<interface>lan</interface>
<protocol>tcp/udp</protocol>
<source>
<network>lan</network>
</source>
<destination>
<any />
<port>22</port>
</destination>
<log />
<descr>traffico ssh</descr>
</rule>
<rule>
```

```
<type>pass</type>
<interface>lan</interface>
<protocol>tcp/udp</protocol>
<source>
<network>lan</network>
</source>
<destination>
<any />
<port>1863</port>
</destination>
<log />
<descr>traffico msn</descr>
</rule>
<rule>
<type>pass</type>
<interface>lan</interface>
<protocol>tcp/udp</protocol>
<source>
<network>lan</network>
</source>
<destination>
<any />
<port>5190</port>
</destination>
<log />
<descr>traffico icq</descr>
</rule>
<rule>
<type>pass</type>
<interface>lan</interface>
<protocol>tcp/udp</protocol>
<source>
<network>lan</network>
</source>
<destination>
<any />
<port>21</port>
</destination>
<log />
<descr>traffico ftp</descr>
</rule>
<rule>
<type>pass</type>
<interface>lan</interface>
<protocol>tcp/udp</protocol>
<source>
<any />
</source>
<destination>
<any />
<port>119</port>
</destination>
<log />
<descr>traffico news</descr>
</rule>
</filter>
<shaper>
<pipe>
<bandwidth>100</bandwidth>
```

```
<descr />
</pipe>
<pipe>
<bandwidth>100000</bandwidth>
<descr />
</pipe>
<pipe>
<bandwidth>2</bandwidth>
<descr />
</pipe>
<rule>
<interface>wan</interface>
<protocol>tcp</protocol>
<source>
<any />
</source>
<destination>
<any />
<port>1-65535</port>
</destination>
<direction />
<iplen />
<tcpflags />
<descr>traffico limitato</descr>
<targetpipe>0</targetpipe>
</rule>
<rule>
<interface>wan</interface>
<protocol>udp</protocol>
<source>
<any />
</source>
<destination>
<any />
<port>1-65535</port>
</destination>
<direction />
<iplen />
<tcpflags />
<descr>traffico limitato</descr>
<targetpipe>0</targetpipe>
</rule>
</shaper>
<ipsec />
<aliases />
<proxyarp />
</m0n0wall>
```

BIBLIOGRAFIA

[1] Andreasson Oskar, "*Iptables Tutorial 1.1.19*", 2001, http://iptables-tutorial.frozentux.net/iptables-tutorial.html

[2] "*Proteggere la rete:Firewall*", http://www.tariffe.it/speciali/speciale_firewall.htm

[3] "*Introduzione ai firewall*", http://queen.rett.polimi.it/~ant/bsd/firewall/

[4] Cheswick William R., Bellovin Steven., Rubin Aviel, "*Firewall e sicurezza di rete*", 2003.

[5] Emiliano Bruni, "*Linux Firewall: metti una marcia in più nella tua rete*", 2002, www.ebruni.it/docs/lf/lf.pdf

[6] Vene.ws, " *firewall-config*", www.vene.ws/firewallconfig/download/firewallconfig.pdf

[7] Carlo Contavalli, "*IPTables for fun –Implementare un firewall in linux* ", 2002, http://www.commedia.it/ccontavalli/docs-it/iptables/iptables4dummies.pdf

[8] Sportolari Francesco, "Reti LAN in pratica: la configurazione", *Linux partico*, 31-36

[9] OpenSkills, "*/etc/dhcpd.conf*", http://openskills.info/infobox.php?IDbox=749&boxtype=path

[10] m0n0wall, "*m0n0wall*", http://m0n0.ch/wall/documentation.php

[11] SmoothWall, "*SmoothWall*", http://www.smoothwall.org/docs/

[12] Anderson james P., "*Monitoring and surveillance*", 1980.

[13] Axelsson Stefan, "*A survay*", 1998.

[14] Axelsson Stefan, "*A survay and taxonomy*", 2000.

[15] Axelsson Stefan, "*Conference on Computer and Comunication Security*", 1999.

[16] Aleph1, "*Smashing the stack for fun and profit*", 1996.

[17] Roesh Martin, "*Leightweight intrusion detection for networks*", 1999.

[18] Jacobson Van, Leres Craing, McCanne Steven "*libpcap*", 1994.

[19] Desai Neil, "*Increasing performance in high speed nids*".

[20] Boyer B., Moore J. S., "*The boyer-moore fast string seaching algorihtm*".

[21] Ivaldi Marco, "SNORT IDS per TUTTI", *Linux&C*, 44-48

[22] OpenSkills, "*Utilizzo di SNORT come NIDS*", http://openskills.info/infobox.php?IDbox=589&boxtype=tips

[22] Shishii, "*HOWTO sull'istallazione e configurazione dell'IDS Snort"*, http://www.shishii.com/dummy/index.php?id=38

[23] OpenSkills, "*ssh-keygen –b 1024 –t dsa*", http://openskills.info/infobox.php?IDbox=168&boxtype=bofh

[24] P2P WatchDog, "*P2P WatchDog*", http://www.p2pwatchdog.com/home.html

[25] P2P Sicuro, "*P2P Sicuro*", http://www.p2psicuro.it/

[26] Benvenuti Christian, "TRAFFIC SHAPING dalla A alla Z", *Linux&C*, 50-56

[27] "*References on CBQ*",
http://www.aciri.org/floyd/cbq.html

[28] "*References on RED*",
http://www.aciri.org/floyd/red.html

[29] INFOSEC, "*Perchè un firewall non è sufficente*",
www.infosec.it/download/Infosecurity2001-Firewall.pdf

[30] Carl Livitt, "*Remotely Exploitable Overflow In mod_mylo For Apache*"
http://www.securiteam.com/unixfocus/5XP0P2AALK.html

[31] Trivedi Chintan, "*Apache Multiple Space Header DoS*"
http://www.securiteam.com/unixfocus/6A0010KBPE.html