UNIVERSITÀ DEGLI STUDI DI CAMERINO

FACOLTÀ DI SCIENZE E TECNOLOGIE

Corso di Laurea in Informatica

Dipartimento di Matematica e Informatica

INTRUSION PREVENTION SYSTEM IMPLEMENTAZIONE E CONFIGURAZIONE

Tesi di Laurea compilativa
In Reti di Elaboratori

Laureando

Luca Pennacchietti

Relatore

Dott. Fausto Marcantoni

ANNO ACCADEMICO 2007 / 2008

Ringraziamenti doverosi vanno a tutti coloro che nel corso di questi tre anni mi hanno accompagnato lungo tutto il percorso di studi. Inoltre ringrazio la mia famiglia che ha reso possibile tutto ció, e i miei amici, che soprattutto negli ultimi mesi non mi hanno fatto mancare l'affetto e il sostegno necessari. Un ringraziamento particolare va al Professor Fuasto Marcantoni per la pazienza e lo spirito con cui ci ha sopportati.

“Sono conscio dello stato della mia ignoranza e pronto a imparare da chiunque, indipendentemente dalla sua qualifica.”

Isaac Asimov

Indice

Introduzione

Dopo la nascita di Internet il modo di comunicare tra le persone è cambiato radicalmente, aggiungendo una nuova dimensione.

Questo cambiamento ha interessato non soltanto la comunicazione tra le singole persone, ma anche il rapporto, di una ditta con la propria clientela.

Oggi, spesso, le ditte e le aziende necessitano per la comunicazione con i propri clienti, di una connessione ad Internet, ed essendoci in ballo capitali è necessario che la connessione alla Rete da parte di queste aziende sia protetta.

Molto spesso le ditte spendono ingenti quantitá di denaro per la realizzazione di sistemi che siano il piú possibile sicuri e protetti da eventuali attacchi provenienti dall'esterno, da parte di malintenzionati.

Attraverso questa Tesi si vuole mostrare come è possibile realizzare un sistema sicuro basato totalmente su software open source.

Esso verrá fornito di firewall e filtraggio del flusso dati, proveniente da fuori e dentro la sottorete creata.

In particolare si mostrerá come evitare intrusioni dall'esterno tramite l'utilizzo di IDS.

I vari programmi installati avranno il compito di rilevare e bloccare i tentativi da parte di malintenzionati, di infiltrarsi all'interno del sistema realizzato. I diversi attacchi verranno catalogati in un database al fine di essere poi riconosciuti in seguito e resi inoffensivi.

Per la configurazione del tutto verranno usate interfacce grafiche che permetteranno all'amministratore di sistema di gestire al meglio e in maniera piú semplice e veloce, la rete realizzata. Saranno piú facilmente consultabili i file di log, all'interno dei quali saranno descritti in maniera quanto piú dettagliata possibile, quegli eventi che minacciano l'integritá di tutto il sistema.

Lo scopo principale di questo lavoro è quello di illustrare il funzionamento di tale sistema, mostrare in maniera approfondita come i software, che sono stati utilizzati, lavorino per ottenere il livello di sicurezza ricercato.

Struttura

La tesi è strutturata in 4 capitoli.

Il primo capitolo riguarda la creazione della rete che servirá per svolgere il lavoro, attraverso la procedura di installazione del sistema operativo, illustrando la corretta configurazione delle due schede di rete si arriverá poi alla descrizione vera e propria della struttura della rete realizzata.

Seguirá poi lo studio approfondito dei vari applicativi, quali DHCP, DNS e Dynamic DNS, imprescindibili per la creazione di tale sistema.

Verrá mostrata a grandi linee anche l'installazione e la configurazione del server proxy e le varie tipologie adottabili, in base alle esigenze dell'utente.

Per ultima cosa avverrá l'installazione sulla macchina di un server web, nel caso in questione Apache.

Il secondo capitolo è la parte centrale e piú importante di questa Tesi, focalizzando l'attenzione sugli aspetti di sicurezza delle reti in generale e piú in dettaglio di quella appena realizzata.

Sará studiato il funzionamento di un firewall nelle sue varie componenti. Quello preso in questione è Netfilter con il tool relativo IPTables, attraverso il quale è possibile una totale interazione con il Firewall.

Proseguendo verranno descritti nei minimi dettagli, programmi che fungono da IDS e IPS, e cioè applicativi in grado di rilevare (IDS) e bloccare (IPS) intrusioni provenienti dall'esterno.

Per questo compito è stato scelto Snort che puó ricoprire entrambi i ruoli:

nella modalitá di base esso si limiterá a rilevare la presenza di minacce, mentre nella modalitá INLine riuscirá anche a bloccare eventuali tentativi di intrusione.

Ovviamente sará illustrata la configurazione di ognuno di essi e ne verranno presentati alcuni esempi.

Nel corso di questo capitolo saranno anche descritte le diverse modalitá di funzionamento dell'IDS utilizzato.

Ci sará anche il riassunto dei cambiamenti da effettuare all'interno dei file di configurazione.

Per il corretto funzionamento di tali applicazioni ne sono necessarie altre, quali un database su cui registrare i diversi avvenimenti rilevati, interfacce grafiche che aiutano l'utente nella gestione del sistema. Anche qui ne verranno mostrati alcuni esempi, anche se per la realizzazione finale del lavoro ne sono state utilizzate soltanto un paio.

Il quarto capitolo comprende tutte le prove e i tentativi che sono stati svolti durante il periodo di realizzazione di questa Tesi, vengono mostrati i vari programmi durante la loro esecuzione e anche la dimostrazione di due attacchi effettuati come test della sicurezza del sistema realizzato.

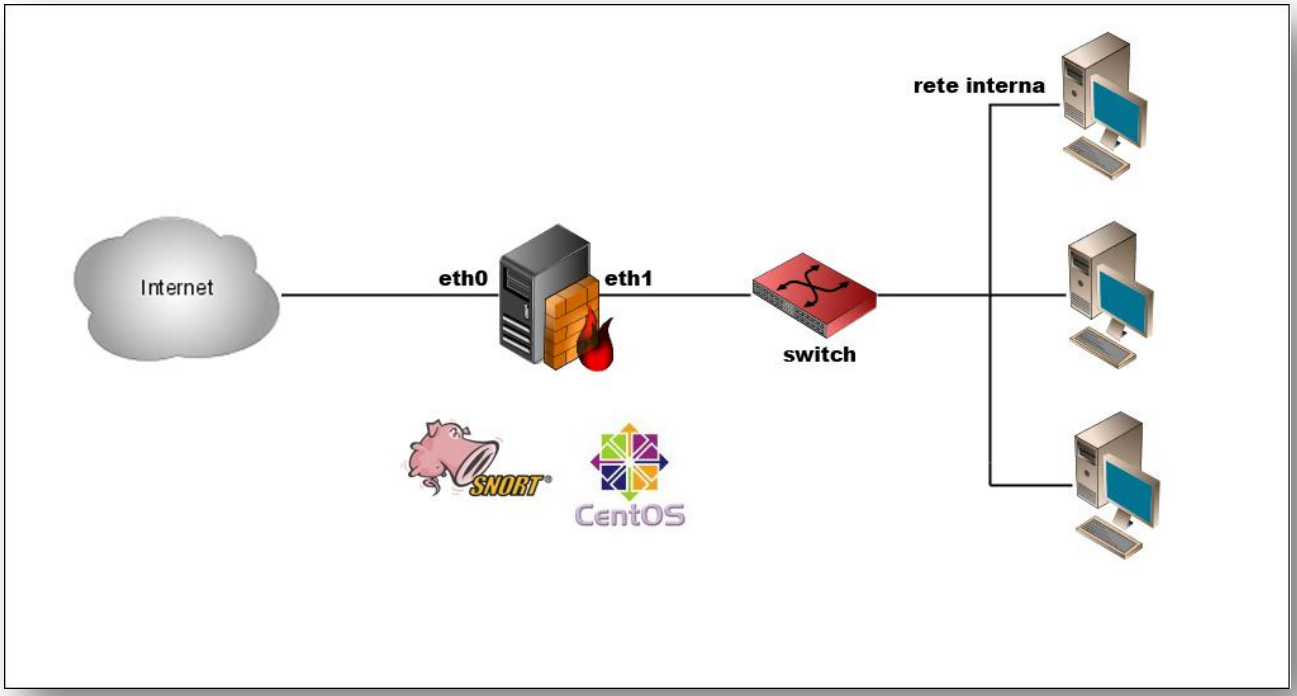
In fine sono presenti gli appendici che comprendono alcuni file di configurazione per intero, allo scopo di illustrare quale sia stato il lavoro svolto per la customizzazione degli applicativi che ne fanno uso.

Inoltre è presente anche una dettagliata bibliografia nella quale sono riportati tutti i link alle pagine web dalle quali è stato tratto il materiale e la documentazione. Vengono anche inseriti i due libri utilizzati per lo studio di ogni componente del progetto realizzato.

Capitolo 1

Realizzazione di una rete

Si vuole realizzare una rete di questo tipo:



Su una macchina vengono installate due schede di rete, rispettivamente eth0 e eth1. Il dispositivo 0 servirá alla macchina per la connessione alla rete esterna, cioè ad internet e quindi per la normale navigazione nel world wide web.

La periferica 1, invece, sará collegata verso una sottorete interna, formata da diverse altre macchine, in questo modo il computer provvisto di due schede di rete diventerá il Server di riferimento per quelli collegati ad essa mediante la eth1.

Essendo quindi il Server della sottorete è necessario, che la macchina, da qui in poi Host1, provveda, per prima cosa, all'assegnamento di un indirizzo IP, con cui identificare ciascun computer presente nella sottorete .

Questo assegnamento puó avvenire in due diversi modi, il primo in maniera statica, nel senso che si puó attribuire a ciascuna macchina un indirizzo fisso, cioè ogni volta che effettuerà l'accesso alla rete, sará sempre identificata con lo stesso IP.

Nell'altro caso, e cioè nel caso in cui si decida di assegnare gli indirizzi ai computer in rete in maniera dinamica, è necessaria la presenza di un Server DHCP. Per fare ció è sufficiente installare sull'Host1 un applicazione che permetta all'Host1 di svolgere tale compito.

Installazione Sistema

Prima di procedere con l' installazione di qualsiasi applicazione deve essere scelto quale sistema operativo adottare per l' Host1.

In questo caso la scelta ricade su CentOS (**C**ommunity **ent**erprise **O**perating **S**ystem).

Si tratta di una distribuzione Linux che deriva da Red Hat Enterprise ed è la soluzione ottimale per coloro che vogliono ottenere velocemente un sistema GNU/Linux di classe superiore.

È composta interamente da software libero e tutto il codice di cui è composta è reso disponibile pubblicamente, a differenza della Red Hat da cui deriva. La Red Hat infatti, sviluppata dalla cAosity Foundation, software house che fornisce soluzioni basate su Linux per Enterprise prodotte da una comunitá, è un sistema a pagamento, rivolto maggiormente ad aziende che necessitano di macchine server.

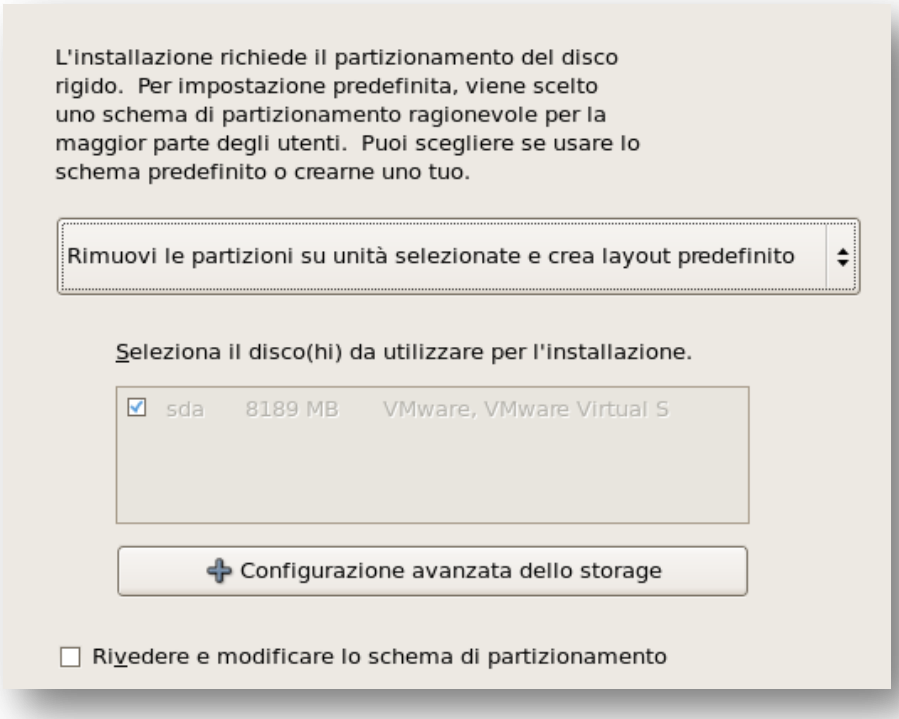
CentOS è reperibile all'indirizzo Internet http://www.centos.org/ , ed è facilmente scaricabile dalla parte inferiore della pagina principale, nella categorie "Release". Sono presenti infatti, per ogni versione del sistema operativo, varie tipologie, dedicate alle varie architetture di processori, a 32 o 64 bit, dedicate in particolare ai server, o distribuzioni live, cioè avviabili senza l'obbligo di installazione, ma direttamente da CD.

Una volta deciso quale delle differenti versioni dell' OS risponde alle proprie necessità, scaricato il file immagine da uno dei server messi a disposizione sul sito e masterizzato su un supporto ottico, è possibile iniziare il procedimento di installazione.

Per prima cosa verranno chieste all'utente diverse informazioni riguardo la configurazione del sistema in sé,come la lingua nella quale sará installato, il layout della tastiera, cioè la disposizione dei caratteri in corrispondenza dei tasti, e infine il fuso orario relativo al paese in cui ci si trova, in modo da poter regolare cosí l'orologio di sistema.

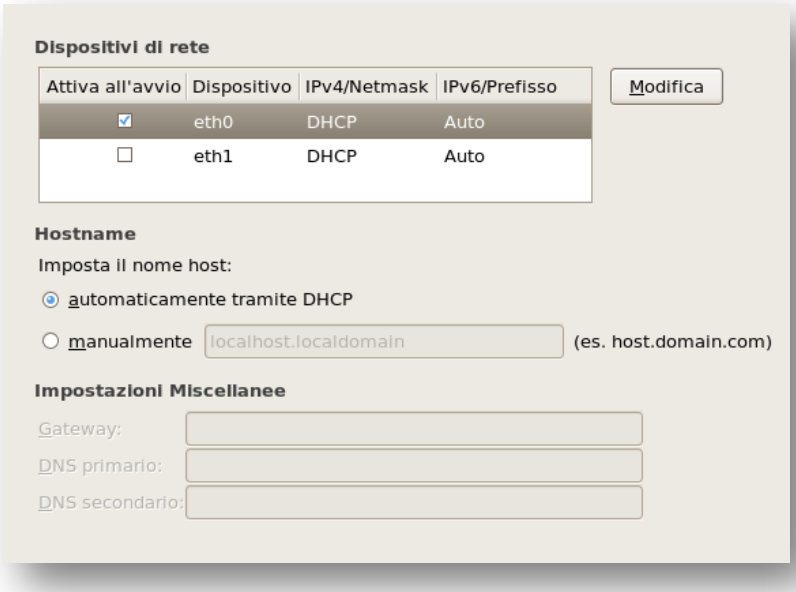
Il passo successivo riguarda la definizione dello spazio dedicato alla partizione in cui verrá installato il sistema.

Il file System che deve essere adottato, avendo a che fare un sistema Linux, sará il tipo EXT3, ricordandosi di riservare un pó di spazio per la partizione dedicata allo SWAP, ossia quella parte di disco fisso utilizzata dal sistema per “appoggiare” i dati una volta esaurita la memoria RAM, piú o meno come avviene con il File Paging sulle piattaforme Windows, con la differenza che sotto Linux viene adibita a tale funzione una vera e propria partizione del disco fisso.

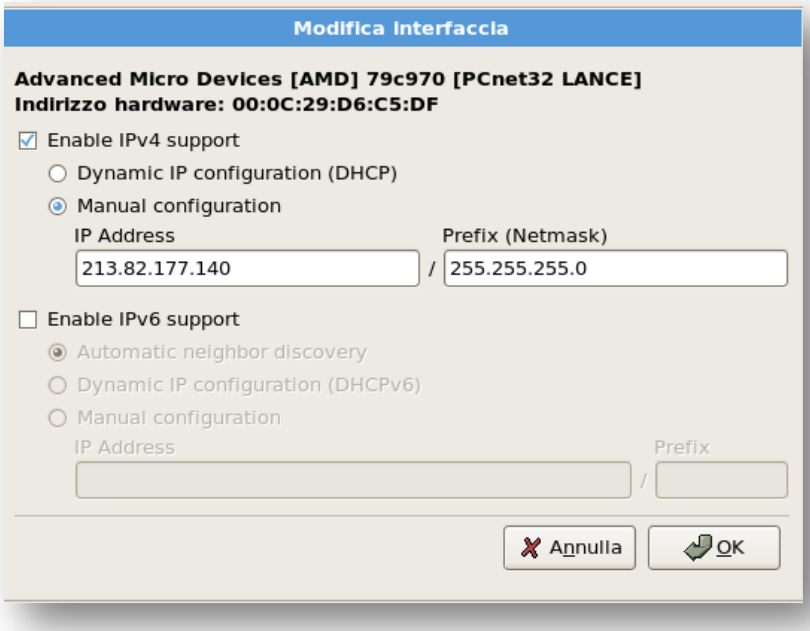


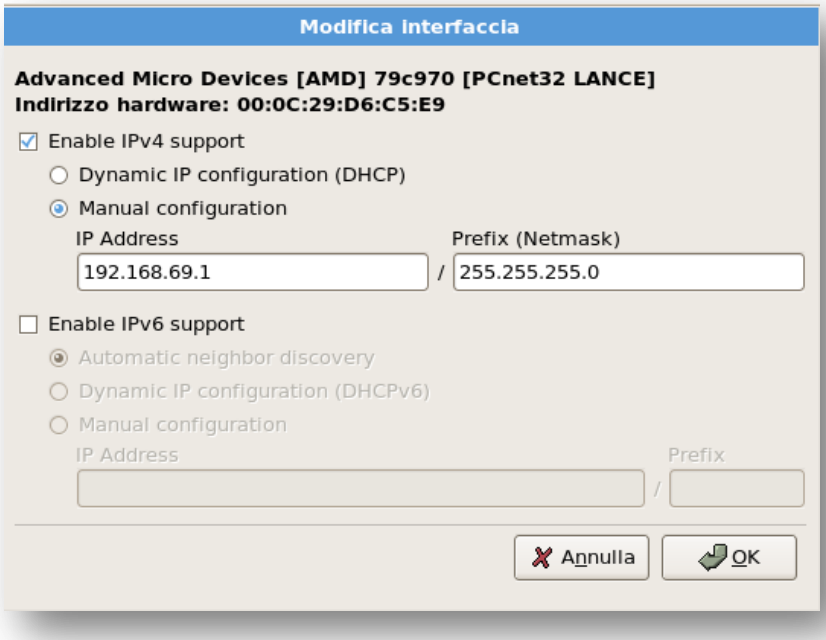
Ora si dovrá configurare una parte fondamentale per la realizzazione del sistema di sicurezza: le schede di rete.

È molto meglio configurare in fase di installazione del sistema le due schede richieste, esse verranno rilevate automaticamente:



Le due periferiche vanno configurate una per una, sotto viene mostrato come:





Nei campi appositi vanno definiti gli indirizzi IP, la eth0 che sará rivolta verso l'esterno avrá come indirizzo :

213.82.177.140

L'indirizzo sará di tipo statico, dato che il dipartimento di informatica ne puó disporre, in caso si dovesse realizzare un sistema simile a questo, si dovrá impostare la periferica 0 con indirizzo dinamico.

Per tutti i casi invece, alla scheda eth1 va assegnato un indirizzo fisso del tipo:

192.168.69.1

Essa verrá collegata alla sottorete interna.

Ora verrá chiesto all'utente di scegliere quale dei due tipi di interfaccia grafica usare. I due disponibili sono KDE e GNOME. In questo caso per maggiore dimestichezza da parte di chi scrive verrá selezionato GNOME, anche se la scelta della GUI è di poca rilevanza ai fini della realizzazione del sistema in questione.

Subito dopo apparirà una finestra simile, nella quale peró sará possibile selezionare dalla lista, i pacchetti che si desidera caricare effettivamente insieme al Kernel.

Attraverso questa schermata è possibile ridurre all'osso la propria configurazione di sistema, evitando tutte quelle applicazioni che si pensa non essere necessarie al lavoro che si andrá a svolgere. Nulla vieta di installare tali pacchetti in seguito, direttamente dal gestore delle installazioni, che in questo caso sará Yum.

Per la realizzazione del sistema di reti sopra descritto si puó fare a meno di alcune applicazioni presenti nella lista. Verranno quindi deselezionati tutti quegl'applicativi riguardanti grafica, documenti di testo e svago, assolutamente superflui nella situazione in questione.

Una volta effettuate queste semplici operazioni partirá l'installazione vera e propria.

In questa fase verranno caricati sul disco, oltre al Kernel del sistema operativo, tutti quei pacchetti che si è deciso di selezionare in precedenza.

Organizzazione della Rete

Come detto prima, per la realizzazione di questo sistema di networking, verranno installate su una macchina due schede di rete per far si che la macchina abbia accesso a internet, e inoltre abbia anche la possibilitá di creare una sottorete collegata ad essa, mediante la seconda periferica.

Lo schema del sistema che si vuole realizzare è il seguente:

L'Host1 tramite la scheda di rete eth1 è collegato alla sottorete attraverso uno switch, che è un dispositivo di rete che inoltra selettivamente verso la porta d'uscita i pacchetti ricevuti, provenienti dalla rete esterna.

Come con un hub, due nodi possono comunicare attraverso uno switch come se questo non ci fosse ma a differenza di quello che farebbe un hub, uno switch normalmente inoltra i pacchetti in arrivo da una delle sue porte, soltanto a quella a cui è collegato il destinatario del frame.

Host1 avrá quindi il compito di filtrare il traffico da e verso la sottorete, svolgendo il ruolo di gateway e firewall.

Le macchine presenti nella rete interna non necessitano di configurazione, a differenza invece del computer Host1, il quale avendo doppia periferica di rete dovrá avere due differenti configurazioni, una per scheda.

L'interfaccia eth0 in questo caso verrá settata con un indirizzo statico 213.82.177.140, la configurazione della Subnet Mask sará 255.255.255.224, il gateway 213.82.177.129 . Per ultimi verranno configurati due DNS, uno 151.99.250.2 e l'altro 151.99.250.2 .

Mentre per la scheda eth1 sará sufficiente settare l' indirizzo IP a 192.168.69.1 e la Subnet Mask a 255.255.255.0 . Da notare che anche in questo caso l'indirizzo IP è statico, ma c'è da dire che questo avverrá soltanto per l' Host1, in quanto le macchine della sottorete saranno soggette ad assegnamento dinamico, tramite DHCP, proprio da parte di Host1.

DHCP

Il DHCP, ossia il Dynamic Host Configuration Protocol, è un protocollo che permette ai dispositivi di rete di ricevere la configurazione dell'indirizzo IP necessario per poter operare su una rete.

In una rete basata sul protocollo IP, ogni calcolatore ha bisogno di indirizzo IP, scelto in modo tale che appartenga alla sottorete a cui è collegato e che sia unico, ovvero non ci siano altri calcolatori che stiano giá usando quell'indirizzo.

Questo sistema toglie agli amministratori di rete il rilevante onere di assegnare manualmente gli indirizzi, soprattutto se si tratta di reti di grandi dimensioni. C'è anche da dire che con l'aumento delle macchine su in internet gli indirizzi IPV4 fissi hanno cominciato a scarseggiare, diminuendo cosí la possibilitá che si possa avere un'indirizzo fisso.

Il protocollo DHCP è formato da diversi componenti:

- Il Client DHCP è un calcolatore che ha bisogno di ottenere un indirizzo IP valido per la sottorete a cui è collegato, ed é anche il programma che si occupa di richiedere l'indirizzo IP e configurarlo.
- Il Server DHCP è il calcolatore che assegna gli indirizzi IP, ed è anche il processo che svolge questa funzione. A volte questa funzione viene svolta da un router.
- Il DHCP Relay è il calcolatore che si occupa di inoltrare le richieste DHCP ad un server, qualora questo non sia sulla stessa sottorete, questo componente è necessario solo se un server DHCP deve fornire molteplici sottoreti. Questa funzione spesso viene implementata in un router.

Il DHCP utilizza il protocollo UDP e le porte usate sono la 67 e la 68 rispettivamente per server e client.

Quando un calcolatore deve ottenere l'indirizzo attiva il processo DHCP client. Esso invia un pacchetto chiamato DHCPDISCOVER in broadcast, con un indirizzo IP sorgente impostato come convenzione a 0.0.0.0, e come destinazione 255.255.255.255. Il pacchetto viene ricevuto da tutti gli host presenti sulla rete e in particolare dal server DHCP presente che puó rispondere o meno con un pacchetto di DHCPOFFER, in cui propone al client un indirizzo IP e altri parametri. Questo

pacchetto è indirizzato direttamente all'indirizzo datalink del client, per cui puó essere inviato solo da un server che si trovi sullo stesso dominio di broadcast.

Se nel dominio ci sono uno o piú DHCP relay, questi inoltrano il pacchetto al loro server di riferimento, che puó rispondere al client sempre attraverso un relay. Il relay agent comunica al server il proprio indirizzo IP sulla sottorete da cui ha ricevuto il pacchetto DHCPDISCOVER, permettendo al server di capire da quale sottorete è arrivata la richiesta, e quindi offrire un indirizzo per la sottorete giusta.

Un server DHCP che debba servire diverse sottoreti IP deve essere configurato per conoscere i parametri di ciascuna.

Il client aspetta un certo tempo di ricevere una o piú offerte, dopodiché ne seleziona una, ed invia un pacchetto di DHCPREQUEST in broadcast, indicando all'interno del pacchetto, con il campo "server identifier", quale server ha selezionato a questo punto il pacchetto raggiunge tutti i server DHCP presenti sulla rete.

Il server che è stato selezionato conferma l'assegnazione dell'indirizzo con un pacchetto di DHCPACK, mentre gli altri server vengono informati che la loro offerta non è stata scelta dal client, e che sulla sottorete è presente un altro server DHCP.

A questo punto, il client è autorizzato ad usare l'indirizzo ricevuto per un tempo limitato, detto tempo di lease. Prima della scadenza di questo tempo , dovrá tentare di rinnovarlo inviando un nuovo pacchetto DHCPREQUEST al server, che risponderà con un DHCPACK se vuole prolungare l'assegnazione dell'indirizzo. Se il client non riesce a rinnovare il suo indirizzo, tornerá allo stato iniziale cercando di farsene attribuire un altro.

Nel sistema in questione l'applicazione scelta per svolgere questo compito è Dhcpd. Disponibile dai repository e qundi facilmente installabile attraverso il comando da terminale:

```
yum install dhcpd
```

Il server andrá poi appositamente configurato attraverso la modifica del file dhcpd.conf, il quale si trova nella directory /etc/dhcpd.conf.

Qui sotto viene riportato un esempio di configurazione di DHCPD per la dichiarazione di una sottorete:

```
subnet 192.168.1.0 netmask 255.255.255.0 {
        option routers                  192.168.1.254;
        option subnet-mask              255.255.255.0;

        option domain-name              "example.com";
        option domain-name-servers       192.168.1.1;

        option time-offset              -5;

# Eastern Standard Time

        range 192.168.1.10 192.168.1.100;
}
```

DNS

Il DNS ovvero il Domain Name System è un servizio utilizzato per la risoluzione di nomi di host in indirizzi IP e viceversa. Il nome denota anche il protocollo che regola il funzionamento del servizio, i programmi che lo implementano, i server su cui questi girano, l'insieme di questi server che cooperano per fornire il servizio.

La possibilitá di attribuire un nome testuale facile da memorizzare a un server migliora di molto l'uso del servizio, in quanto agli esseri umani risulta molto piú facile ricordare nomi testuali piuttosto che numeri.

La risoluzione inversa è utile per identificare l'identitá di host, o per leggere il risultato di un traceroute.

Per utilizzare il servizio, è necessario configurare su ciascun client uno o più server DNS di riferimento.

Quando un sistema ha la necessità di comunicare con un altro sistema, chiede al server DNS di riferimento di effettuare il processo detto di "risoluzione" del nome in un indirizzo IP. Il server effettua una ricerca all'interno del suo database per ottenere l'indirizzo IP corrispondente al sistema ricercato.

Se il server interrogato possiede l'informazione richiesta, il processo di ricerca termina con l'invio dell'indirizzo IP al richiedente. Se la ricerca ha esito negativo il server effettua una richiesta "ricorsiva".

Il DNS utilizza il protocollo di trasporto UDP e la porta 53 per soddisfare le richieste di risoluzione provenienti dagli host.

I server DNS effettuano gli zone transfer usando il protocollo di trasporto TCP e la porta 53.

Il lato client del servizio DNS è normalmente implementato tramite librerie di sistema, che spesso lo integrano con altri servizi di risoluzione, o con la consultazione di file locali, in modo che un utente possa utilizzare un nome simbolico in un'applicazione ed ottenere la sua risoluzione in un indirizzo IP senza preoccuparsi di quale strumento è stato utilizzato per ottenere la risoluzione.

I DNS dividono le sottoreti in diversi spazi e in maniera gerarchica, assegnando ad essi dei nomi. Questi spazi vengono definiti “zone” .

Una zona è un sottoinsieme dello spazio dei nomi definito dal DNS, contenente dei domini (e eventualmente dei sottodomini), gestiti tutti all’unisono. Un singolo server DNS può essere configurato per gestire una o più zone; ogni zona è per così dire "appesa" a un nodo specifico dell’albero che sostituisce la struttura del DNS.

Dynamic-DNS

Il termine DNS dinamico, o DDNS, indica un insieme di tecnologie che permettono di inserire automaticamente in una zona DNS gli indirizzi di calcolatori che ottengono un indirizzo non statico, di solito questo avviene attraverso il protocollo DHCP o PPP.

In una rete locale, questa funzionalità può essere utilizzata direttamente dai client, o può essere configurata usando apposite applicazioni.

Il DDNS viene inoltre utilizzato da servizi commerciali per permettere agli utenti dial-up (modem, ADSL) di registrare un nome corrispondente all'indirizzo che viene loro assegnato di volta in volta dal loro provider. In questo modo, un host con indirizzo IP dinamico è sempre raggiungibile. Esistono client DDNS sia sotto forma di applicazioni che all'interno di router.

Il servizio di DNS dinamico è costituito da una serie di client dinamici, da uno o piú server e da un protocollo di comunicazione tra le due parti. Quando un client dinamico ottiene un indirizzo IP, contatta uno dei server e lo informa del suo IP attuale. Il server inserisce allora un record DNS che punta al nuovo indirizzo del cliente. In questo modo, gli altri host sono in grado di ottenere l'indirizzo IP attuale del client dinamico utilizzando il normale servizio DNS, e quindi senza essere a conoscenza del fatto che l'host che contattano ha un indirizzo registrato dinamicamente.

Il protocollo DNS è implementato da diversi software. Quello che verrá utlizzato per la realizzazione del sistema preso in considerazione sará Bind.

Bind è l'acronimo di Berkeley Internet Name Domain, ed è il server DNS piú usato su internet, specialmente sui sistemi basati su Unix, sui quali è lo standard di fatto.

Esso è un pacchetto software composto principalmente da un Domain Name Service server chiamato Named, da una libreria per la risoluzione dei nomi e da alcune utility per la configurazione e alcuni tool per la gestione del servizio.

La prima apparizione di Bind sulla rete risale ai primi periodi dell'esistenza di internet, quando i primi server erano gestiti proprio con questa applicazione.

Esso peró presentava grossi problemi di sicurezza, che in seguito hanno portato alla sua completa riscrittura. La sua ultima versione è compatibile con le evoluzioni del protocllo DNS, oltre a incorporare nuove funzionalità come estensioni per la sicurezza, compatibilitá con IPv6 e supporto per i sistemi multiprocessore.

Per l' installazione del programma é sufficiente inserire da linea di comando la stringa:

```
yum bind bind-chroot
```

In questo modo verrá installato il programma e anche l'applicazione chroot, che permette di installare Bind in un percorso nascosto, cioè isolato dal resto del file system. Cosí facendo Bind potrá modificare soltanto i file riguardanti se stesso, senza interferire con il resto, ma alla stessa maniera, se un attacco proveniente dall'esterno dovesse arrivare all'interno della directory chroot non potrebbe fare altro che modificare i file di Bind, senza poter intaccare il resto del sistema.

La configurazione di Bind, versione 9.x.x viene gestita dal file named.conf generalmente posto in /etc o in /etc/bind. Questo file che è un normale file di testo ASCII contiene direttive e commenti. Specifica inoltre i file delle zone e la loro ubicazione nel filesystem.

Proxy

Il proxy è un dispositivo che si interpone tra un server e il client, inoltrando le richieste e le risposte dell'uno e dell'altro. Il client si collega al proxy che provvedere a contattare il server per assecondare le richieste del client. Anche viceversa riceverá la risposta del server e la inoltrerà al client. A differenza di bridge o router, il proxy lavora a livello applicativo gestendo un numero limitato di protocolli applicativi.

Un proxy ha diversi scopi:

- mantenere la macchina in anonimato
- accelerare l'accesso alle risorse (via caching).
- Filtrare i contenuti

Un server proxy che riceve le richieste e le inoltra inalterate è chiamato di solito Gateway o a volte tunneling proxy..

Caching proxy server

Un server proxy che usa il caching accelera le rechieste recuperando i contenuti salvati da una richiesta precedente fatta dallo stesso client o da qualsiasi altro client.

Conservando in locale copie delle richieste, permette di ridurre in maniera significativa il consumo della banda e di conseguenza si ha un significativo aumento delle prestazioni. La maggior parte dei provider e delle grandi imprese usano un proxy caching. Queste macchine sono costruite per offrire alti livelli di prestazioni in ambito di file system, implementando spesso RAID e journaling.

Un altro importante utilizzo del server proxy è quello di ridurre i costi hardware. Un'organizzazione uó avere molti sistemi sulla stessa rete o sotto il controllo di un singolo server, vietando la possibilitá di una connessione ad internet per ciascun sistema. In questo caso i sistemi vengono collegati a un server proxy collegato a sua volta al server principale.

Web proxy

Un proxy che concentrato sul traffico www è chiamato "web proxy".

L'uso piú comune di web proxy è di essere utilizzato come una web cache.

La maggior parte dei programmi di proxy, ad esempio Squid, fornisce la possibilitá di negare l'accesso ad alcuni URL, contenuti in una lista nera, fornendo cosí il filtraggio del contenuto. Ció viene di solito usato in ambiente aziendale, anche se con il crescente uso di Linux questa funzione non è piú limitata alle grandi imprese ma viene utilizzata anche nelle piccole imprese e nelle abitazioni.

In alcuni casi il proxy web viene utilizzato per formattare le pagine web per uno scopo specifico, come ad esempio per le pagine web lette dai cellulari o da PDA.

Content-filtering web proxy

Un web proxy server con content-filtering fornisce il controllo amministrativo sul contenuto.

Questo tipo di filtraggio dei contenuti viene utilizzato in ambito commerciale e non (in particolare nelle scuole) per garantire che l'utilizzo di internet sia conforme alla politica di un uso accettabile della rete.

Alcuni dei piú comuni metodi utilizzati per il filtraggio dei contenuti comprendo :

- URL o blacklist DNS
- Filtraggio URL regex
- Filtraggio MIME
- Filtraggio del contenuto con parole chiave

Un content-filtering proxy viene spesso supportato da un autenticazione degli utenti, per il controllo dell' accesso al web. Spesso il proxy produce anche log, per fornire informazioni dettagliate su gli accessi ad URLs da parte degli utenti, o per monitorare l'uso della banda.

Puó collaborare con software antivirus per fornire protezione contro virus e malware, attraverso la scansione del traffico in entrata, in tempo reale.

Anonimizzare una connessione mediante un proxy

Un anonimous proxy server di solito tenta di anonimizzare una connessione. Ció è facilmente aggirabile da degli amministratori di rete, e quindi inutilizzabile.

Ci sono diversi tipi di "anonymizer". Una delle varianti piú comuni è l'open proxy, e poiché difficilmente rintracciabile, è utile per coloro che cercano l'anonimato online.

Utilizzando un server proxy per ottenere l 'anonimato comporta peró dei rischi.

Infatti tutti i dati che vengono inviati passano attraverso il proxy, e la maggior parte delle volte in forma non criptata. Si tratta quindi di un possibile rischio per il fatto che il server possa registrare tutto ció che gli è stato inviato, compresi dati d'accesso e password in chiaro.

La linea di principio da utilizzare è quella di essere molto cauti con l'utilizzo di server proxy per l'anonimizzazione, e ovviamente si consiglia solo l'utilizzo di proxy affidabili e si sconsiglia di evitare quelli dalla integritá non appurata. In caso di impossibilità di non utilizzo di tali server sconosciuti evitare quantomeno l'utilizzo di password e informazioni riservate.

Installazione server web

I server web sono programmi che consentono di distribuire su internet pagine HTLM.

Essi restano in ascolto delle richiseste di accesso a un sito web, le processa e restituisce dei dati come risposta. Tali informazioni, che contengono tutti gli elementi necessari per la visualizzazione di una pagina web composta di testi ed immagini, vengono analizzate da parte del browser nel miglior modo possibile quindi presentate all'utente.

I server web comunicano con il browser Internet installato sul personal computer "client", cioè la macchina dalla quale si effettua l'accesso, attraverso il protocollo http.

Esso standardizza il processo di invio e di ricezione dei dati cosicché qualsiasi client possa agevolmente comunicare con qualunque tipo di server web, senza problemi di compatibilitá.

Il funzionamento di un web server, nel caso piú semplice, è la trasmissione di pagine HTLM, questo meccanismo funziona piú o meno cosí:

- Il browser richiede al server una pagina HTLM
- Il server recupera la pagina HTLM e la spedisce al browser
- Il browser richiede altre risorse contenute nella pagina HTLM
- Il server fornisce queste risorse al browser che visualizza la pagina.

Le capacità di un server possono tuttavia essere incrementate mediante l'utilizzo di applicazioni server-side, cioè programmi che vengono eseguiti direttamente sul server.

Apache

Apache HTTP Server, o più comunemente è il nome dato alla piattaforma server Web più diffusa, in grado di operare da sistemi operativi UNIX-Linux e Microsoft.

É un software che realizza le funzioni di trasporto delle informazioni, di internetwork e di collegamento, ha il vantaggio di offrire anche funzioni di controllo per la sicurezza come quelli che compie il proxy.

È il server web più utilizzato al mondo, nato per funzionare come processo "stand alone" ossia senza richiedere l'appoggio di altre applicazioni o di altri elementi software.

Essendo Apache completamente gratuito è possibile scaricare dal sito, httpd.apache.org, oltre che all'applicazione, disponibile sia la versione per Linux sia la versione per piattaforme Windows, anche i codici sorgenti del programma.

Apache è composto da un demone, in ambiente Unix e da un servizio in ambiente Microsoft.

L'installazione di Apache è molto semplice in quanto è sufficiente aprire un terminale e digitare il comando:

```
Yum install httpd
```

Se il computer è connesso in rete, in pochi secondi saranno scaricati i pacchetti necessari e installati immediatamente.

La configurazione del web server da parte degli amministratori del sito può essere effettuata tramite il file httpd.conf, che sui sistemi Unix solitamente è messo sotto /etc/http/conf, mentre sui sistemi Windows è situato nella sottodirectory conf della cartella di installazione di apache.

Attraverso le impostazioni contenute nel file httpd.conf è possibile accedere a uno o più siti, gestendo varie caratteristiche di sicurezza ed estensioni per le pagine attive e dinamiche.

La compilazione dei sorgenti invece si completa effettuando una serie di passi:

```
./configure
make
make install
```

Le opzioni permettono di abilitare il supporto di specifiche funzionalità o di definire parametri particolari.
Opzioni comuni per ./configure:

`--prefix` - Imposta la directory di base in cui installare tutti i file

`--show-layout` - Mostra, senza compilare, il layout della disposizione dei vari file

`--enable-layout=`*`layout`* - Imposta un layout predefinito adeguato al proprio sistema (Apache, RedHat, GNU...)

`--enable-suexec` - Abilita il supporto SuExec

`--with-ssl=`*`/path/lib/ssl`* - Abilita il supporto SSL (necessario mod_ssl)

`--enable-mods-shared=`*`most`* - Compila e attiva come moduli runtime la maggior parte di quelli forniti con i sorgenti. Oltre a "most" si può scrivere "all" oppure l'elenco, separato da uno spazio e fra virgolette, dei nomi dei moduli.

`--enable-modules=`*`most`* - Stessa sintassi del precedente, ma abilita la compilazione di moduli statici.

Per verificare il giusto funzionamento di Apache è possibile eseguire una sorta di test:

```
cd /var/wwwhtlm

wget http://wwwinternetsecurityguru.com/index.php.txt

mv index.php.txt index.php
```

Capitolo 2

Sicurezza delle reti

Dal momento in cui si interconnettono più computer tra loro nasce il problema della sicurezza della rete. Una rete, infatti, è soggetta a diversi tipi di vulnerabilità, che possono essere sfruttate da terzi per intercettare il traffico dei dati.

Nei primi anni dell'esistenza delle reti, esse erano utilizzate prevalentemente da ricercatori universitari per inviare e ricevere email, e dalle aziende per condividere stampanti. In quelle applicazioni non si prestava particolare attenzione alla sicurezza. Oggi milioni di persone usano le reti per fare acquisti, lavorare con la banca, quindi la sicurezza delle reti è diventato un problema molto rilevante.

La maggior parte dei problemi di sicurezza sono causati da persone malintenzionate che tentano di danneggiare in diversi modi gli altri utenti della rete.

I problemi di sicurezza delle reti si possono orientativamente dividere in quattro aree interconnesse tra loro:

- Segretezza
- Autenticazione
- Nonrepudation
- Controllo integrità

La segretezza, detta anche confidenzialità, si occupa di mantenere le informazioni fuori dalla portata degli utenti non autorizzati. Questo rappresenta l'idea di sicurezza che di solito hanno le persone.

L'autenticazione si occupa di stabilire l'identità del soggetto con cui si sta comunicando, prima di rivelare informazioni sensibili o concludere transazioni commerciali etc.

Il controllo dell'integrità dei dati o oggetti si effettua per verificare che non vi sia stata manomissione, o modifica o addirittura la distruzione di parte o della totalità di essi.

Per garantirla si deve avere la fiducia nelle persone autorizzate alle modifiche e la protezione da parte di estranei o persone no autorizzate.

Per i sistemi sulle reti, in particolare su Internet, le minacce da cui difendersi sono molteplici e di diversa entità:

- Virus, worm e altri agenti automatici, che si diffondono autonomamente su sistemi non aggiornati, senzá la volontà e l'intervento umano e posso causare danni di entità variabili.
- Scan su larga scala alla ricerca indiscriminata di sistemi vulnerabili, sfruttando buchi di sicurezza generalmente noti. Questi scan possono essere effettuati da diversi tipi di attackers, come script kiddies o cracker. Questi ultimi maggiormente pericoli, poiché a conoscenza di tecniche di attacco molto sofisticate.
- Attacchi mirati determinati dalla penetrazione di uno o più host di un'entità specifica per modalità varie. Sono insidiosi e i più pericolosi, perché se effettuati da cracker capaci e determinati possono compromettere anche sistemi ben protetti.

Le protezioni e quindi le policy di sicurezza da adottare hanno diversi livelli di complessità. Se è relativamente semplice difendersi dai primi due tipi di minacce, impostando una struttura che espone solo i servizi strettamente necessari e tenendo aggiornato il software che gestisce questi servizi, diventa molto più impegnativo realizzare una struttura estremamente sicura, in grado di resistere agli attacchi più smaliziati, sia tramite internet che per vie indirette.

Non esistono certezze nel campo dell'Information Security, non esiste un sistema in rete sicuro al 100%: di un software che oggi appare sicuro, domani potranno essere trovati buchi sfruttabili per intrusioni esterne; una rete per quanto protetta deve essere amministrabile e chi la amministra o la utilizza con privilegi particolari, può essere esso stesso una minaccia.
Per quanto le problematiche di sicurezza siano quindi vaste e abbiano fattori diversi, si possono definire alcune linee guida minime per disegnare una rete il più possibile protetta da quelle che, di fatto, sono le maggiori minacce: l'intrusione di ignoti da Internet.
L'accesso da remoto a un sistema può avvenire quasi esclusivamente tramite una porta TCP o UDP aperta su un host in rete, per cui a livello di networking il controllo e l'attenzione vanno posti su quali porte lasciare accessibili e come.

Realizzare una rete con un design di base sicuro richiede l'applicazione di alcune semplici regole di base, che vanno adattate ai diversi contesti specifici:

- **Chiudere le porte su IP pubblico non utilizzate**
  Ogni tentativo d'intrusione remoto può avvenire solo tramite eventuali porte aperte su un host. La prima, fondamentale, comoda e utile procedura da adottare per iniziare a proteggere il proprio network è quella di non lasciare varchi inutili ai potenziali intrusi.
  La chiusura delle porte può essere fatta sia a livello dei singoli host, rimuovendo tutti i servizi che non servono, sia a livello di firewall perimetrale, con un rigoroso packet filtering.
  Le due soluzioni non sono alternative, anzi possono tranquillamente coesistere: sull'host si rimuovono i servizi inutili, sul firewall si possono gestire quali indirizzi IP sorgenti possono accedere a determinati servizi.
  Questo si ricollega al secondo punto chiave:

- **Centralizzare i punti di controllo del traffico**
  Tipicamente una rete aumenta la sua complessità con la sua naturale crescita, ma se non viene strutturata dall'inizio in modo lungimirante, rischia di diventare una matassa di router e firewall su cui si deve intervenire in caso di problemi o "aperture di porte".
  Centralizzare i punti di routing e firewalling aiuta a implementare più facilmente modifiche alle configurazioni esistenti e a diagnosticare problemi di rete, oltre a ridurre i *point of failure* dell'infrastruttura.
  Il firewall perimetrale dovrebbe, di default, bloccare ogni tipo di pacchetto dall'esterno e prevedere regole specifiche per ogni server in produzione. Queste regole possono essere di due tipi:

  1. Regole che aprono una determinata porta di un dato host a tutta Internet. Queste sono quelle necessarie per i servizi pubblici in produzione (server web, server di posta, DNS ecc.) ed è bene che "aprano" sono le porte strettamente necessarie e non tutto un indirizzo IP (anche se su questo risponde una macchina con tutti i servizi inutili disattivati). Sui firewall che gestiscono regole di filtraggio in modo sequenziale (le catene di iptables, le ACL del Cisco IOS o del PIX...) è sempre meglio mettere per prime le regole che riguardano i flussi di traffico maggiori, per diminuire l'impatto sulle risorse evitando inutili controlli su regole di filtering poco

usate.

2. Regole che aprono l'accesso a determinate porte e IP da specifici IP sorgenti. Possono essere necessarie per aprire l'accesso FTP a dei data feed provider o permettere l'accesso in VPN da una determinata sede o l'accesso ad un database da un server remoto. Queste tipicamente tendono a rendere più verbosa e complicata la configurazione di un firewall, dal momento che devono adattarsi a IP sorgenti specifici, ma visto che sono necessarie è opportuno centralizzarle.
   Nell'applicare simili regole, che tendono per il loro numero ad appesantire il firewall, si consiglia buon senso e tendenza al raggruppamento: se ci sono più IP da "aprire" della stessa subnet, si può considerare l'apertura dell'intera subnet, se ci sono più porte da aprire (sopratutto porte che di fatto permettono l'accesso al sistema, come telnet e ssh) può aver senso aprire l'accesso all'intero IP dell'host di destinazione, senza specificarne le singole porte.

- **Controllare tutti i punti di accesso**
  In termini di sicurezza informatica, l'anello debole, quello meno controllato e sicuro, riduce la sicurezza di tutta l'infrastruttura. Per questo motivo è fondamentale identificare e ponderare ogni punto di accesso ai nostri sistemi da remoto. Questo include router e linee di partner, clienti e fornitori, macchine per permettere l'accesso via modem alla rete da proteggere.
  A prescindere quindi da tutte le protezioni del caso da prendere sia a livello fisico, che a livello dei client degli utenti (sviluppatori, sistemisti, redattori ecc) che in qualche modo possono accedere in modo privilegiato alla nostra rete, vanno verificati anche tutti i punti di entrata accessibili pubblicamente.
  Come sempre, se questi prevedono un accesso via password, la coppia login-password dovrebbe non essere facilmente intuibile e il relativo traffico dovrebbe essere criptato.

- **Design multilayer in caso di reti complesse**
  Se la rete da gestire comprende decine di host, di diversi utilizzi (macchine in produzione di front-end, cioè che devono interagire direttamente con Internet, macchine di backend come database o sistemi di monitoring, macchine di stage e pre-produzione, macchine di sviluppo, client di sistemisti e sviluppatori ecc) è opportuno valutare una suddivisione della nostra

rete in più livelli, cercando di proteggere quelli più interni e delicati (il backend) e limitando allo stretto necessario l'accesso a queste macchine da parte di altre macchine più esposte (per esempio i server pubblici di front-end, i server di sviluppo, i client vari).
In strutture complesse, inoltre, le macchine di frontend, se possibile, non dovrebbero avere accesso in scrittura ai dati: un web server per esempio non dovrebbe poter scrivere sulla directory in cui sono contenute le sue pagine web, eventualmente generate da un Content Management System sul backend.
Con reti più semplici è comunque buona norma cercare di limitare allo stretto necessario le possibilità di comunicazione di una macchina con qualsiasi altra macchina del network, in modo da limitare i potenziali danni su tutti i sistemi dopo l'intrusione in uno degli host.

- **Diversificare le difese**

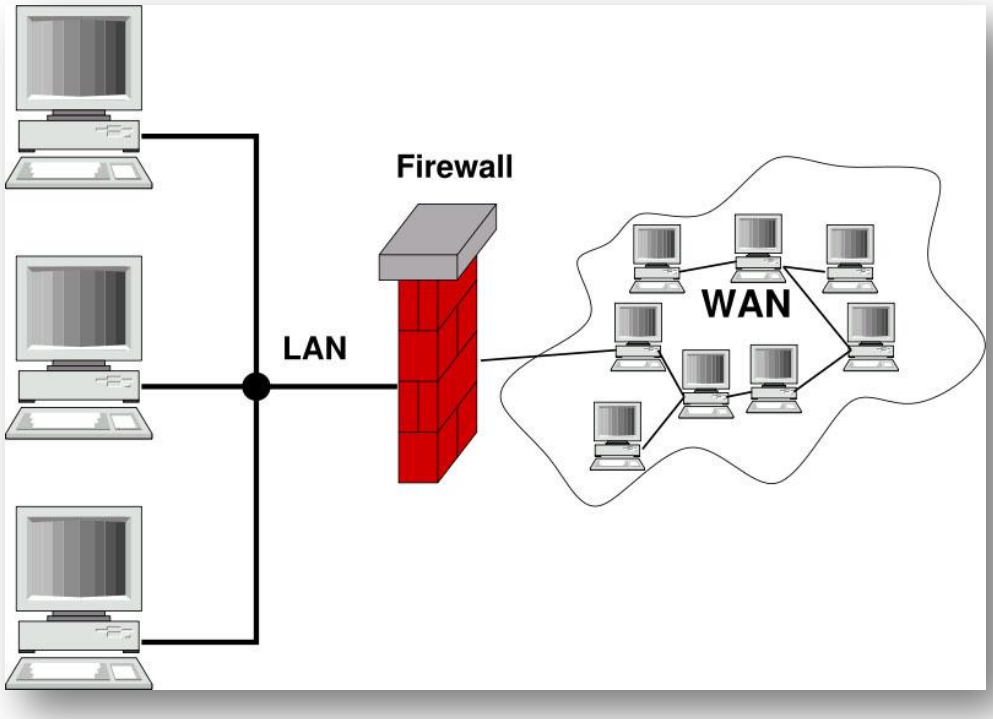
  La diversificazione delle difese è un aspetto molto importante per quanto riguarda la configurazione di un sistema che sia discretamente sicuro : non basta il firewall, non basta l'IDS, non basta la password sul Bios: ogni possibile mezzo per rendere difficile la vita ad un cracker è un buon mezzo, anche se a volte rende la vita difficile anche ad un sysadmin.
  Un esempio tipico è la rimozione di ogni programma non strettamente necessario da un server in produzione: nessun comando per scaricare un file da remoto (wget, ftp, lynk, irc, curl ecc), nessun compilatore che permetta a un intruso di compilarsi i suoi sorgenti maligni direttamente sul sistema, sistema di Mandatory Access Control che impedisca ad un qualsiasi processo di fare qualcosa di diverso da quello che è stato configurato per fare.
  Dal punto di vista della sicurezza sono tutti buoni mezzi che aumentano la solidità del sistema e al contempo ne aumentano la difficoltà di gestione.
  A queste indicazioni generali, che si applicano considerando solo a livello di rete le problematiche di sicurezza, si affiancano tutte le altre procedure volte a rendere sicuri i servizi che si lasciano aperti in rete.
  Come si è potuto intravedere, quindi, realizzare un sistema estremamente sicuro è complicato e richiede sforzi e risorse notevoli, ma realizzare una struttura di rete "ragionevolmente" sicura, che ci possa proteggere dalla maggior parte delle insidie non è eccessivamente gravoso e si può riassumere, semplificando all'estremo, in tre principi fondamentali:
  - ridurre al massimo le porte esposte a Internet;

- avere software aggiornati in ascolto sulle porte esposte;
- non configurare in modo scorretto i servizi pubblici.

Firewall

I firewall, letteralmente muro di fuoco, sono una versione moderna del vecchio rimedio medievale per le emergenze di sicurezza: scavare un fossato profondo intorno al proprio castello. Questo sistema obbligava chiunque volesse entrare o uscire dal castello a passare per un singolo ponte levatoio, dove i soldati poteva facilmente eseguire le sue ispezioni.

In informatica un firewall è un componente passivo di difesa perimetrale che può anche svolgere funzioni di collegamento tra due o più tronconi di rete. Usualmente la rete viene divisa in due sottoreti: una detta esterna, comprende l'intera Internet mentre l'altra interna, detta LAN, comprende una sezione più o meno grande di un insieme di computer locali.



Schema di una sottorete protetta da un firewall.

Il Firewall è un apparato di rete hardware o software che ha il compito di filtrare tutti i pacchetti entranti e uscenti allo scopo di proteggere gli host, applicando una serie di regole che contribuiscono a rendere sicuro il sistema.

In realtà un firewall può essere realizzato con un normale computer munito di almeno due schede di rete, oppure può essere una funzione inclusa in un router o può essere un apparato specializzato. Esistono anche i cosiddetti firewall personali, che sono programmi, installati sui normali calcolatori, che filtrano solamente i pacchetti che entrano ed escono dal quel calcolatore, in tal caso viene utilizzata una scheda di rete soltanto.

La funzionalità principale in sostanza è quella di creare un filtro sulle connessioni entranti ed uscenti, in questo modo il dispositivo innalza il livello di sicurezza della rete e permette sia agli utenti interni che a quelli esterni di operare nel massimo della sicurezza. Il firewall agisce sui pacchetti in transito da e per la zona interna potendo eseguire su di essi operazioni di:

- Controllo
- Modifica
- Monitoraggio

Questo grazie alla sua capacità di "aprire" il pacchetto IP per leggere le informazioni presenti sul suo header, e in alcuni casi anche di effettuare verifiche sul contenuto del pacchetto.

Oltre al firewall a protezione perimetrale ne esiste un secondo tipo, definito "*Personal Firewall*", che si installa direttamente sui sistemi da proteggere (per questo motivo è chiamato anche Firewall Software). In tal caso, un buon firewall effettua anche un controllo di tutti i programmi che tentano di accedere ad Internet presenti sul computer nel quale è installato, consentendo all'utente di impostare delle regole che possano concedere o negare l'accesso ad Internet da parte dei programmi stessi, questo per prevenire la possibilità che un programma malevolo possa connettere il computer all'esterno pregiudicandone la sicurezza.

Il principio di funzionamento differisce rispetto a quello del firewall perimetrale in quanto, in quest'ultimo, le regole che definiscono i flussi di traffico permessi vengono impostate in base all'indirizzo IP sorgente, quello di destinazione e la porta attraverso la quale viene erogato il servizio, mentre nel personal firewall all'utente è sufficiente esprimere il consenso affinché una determinata applicazione possa interagire con il mondo esterno attraverso il protocollo IP.

Tipologie di firewall, in ordine crescente di complessità:

- Il più semplice è il packet filter, che si limita a valutare gli header di ciascun pacchetto, decidendo quali far passare e quali no sulla base delle regole configurate. Ciascun pacchetto viene valutato solamente sulla base delle regole configurate, e per questo un firewall di questo tipo è detto anche stateless. Alcuni packet filter, analizzando i flag dell'header TCP, sono in grado di discriminare un pacchetto appartenente a una "connessione TCP stabilita (established)" rispetto a quelli che iniziano una nuova connessione, ma non sono in grado di riconoscere un pacchetto malevolo che finga di appartenere ad una connessione TCP stabilita. Molti router posseggono una funzione di packet filter.
- Un firewall di tipo stateful inspection, tiene traccia di alcune relazioni tra i pacchetti che lo attraversano, ad esempio ricostruisce lo stato delle connessioni TCP. Questo permette ad esempio di riconoscere pacchetti TCP malevoli che non fanno parte di alcuna connessione. Spesso questo tipo di firewall sono in grado anche di analizzare i protocolli che aprono più connessioni (ad esempio FTP), inserendo nel payload dei pacchetti informazioni di livello rete e trasporto, permettendo così di gestire in modo puntuale protocolli di questo tipo.
- I firewall di tipo deep inspection effettuano controlli fino al livello 7 della pila ISO/OSI, ovvero valutano anche il contenuto applicativo dei pacchetti, ad esempio riconoscendo e bloccando i dati appartenenti a virus o worm noti in una sessione HTTP o SMTP.
- I cosiddetti Application Layer Firewall sono apparati che intercettano le connessioni a livello applicativo. A questa categoria appartengono i proxy. In tali casi, la configurazione della rete privata non consente connessioni dirette verso l'esterno, ma il proxy è connesso sia alla rete privata che alla rete pubblica, e permette alcune connessioni in modo selettivo, e solo per i protocolli che supporta.

La sintassi della configurazione di un firewall in molti casi è basata su un meccanismo di lista di controllo degli accessi (ACL), che possono essere statiche (quindi modificabili solo tramite configurazione esplicita) o dinamiche (cioè che possono variare in base allo stato interno del sistema, come ad esempio nel Port knocking).

Una funzione spesso associata al firewall è quella di NAT (traduzione degli indirizzi di rete), che può contribuire a rendere inaccessibili i calcolatori sulla rete interna.

Molti firewall possono registrare tutte le operazioni fatte (logging), effettuare registrazioni più o meno selettive (ad esempio, registrare solo i pacchetti che violano una certa regola, non registrare più di N pacchetti al secondo), e tenere statistiche di quali regole sono state più violate.

La registrazione integrale dell'attività di un firewall può facilmente assumere dimensioni ingestibili, per cui spesso si usa il logging solo temporaneamente per diagnosticare problemi, o comunque in modo selettivo (logging dei soli pacchetti rifiutati o solo di alcune regole). Tuttavia, l'analisi dei log di un firewall (o anche dei contatori delle varie regole) può permettere di individuare in tempo reale tentativi di intrusione.

Talvolta a un firewall è associata anche la funzione *rilevamento delle intrusioni (IDS)*, un sistema basato su euristiche che analizza il traffico e tenta di riconoscere possibili attacchi alla sicurezza della rete, e può anche scatenare reazioni automatiche da parte del firewall (Intrusion prevention system).

Netfilters/IPTables

Netfilter è un componente del kernel del sistema operativo Linux, che permette l'intercettazione e manipolazione dei pacchetti .

iptables è il programma che permette agli amministratori di sistema di configurare netfilter, definendo le regole per i filtri di rete e il reindirizzamento NAT. Spesso con il termine iptables ci si riferisce all'intera infrastruttura, incluso netfilter.

Netfilter trova applicazione sia in calcolatori che vengono usati come host che per realizzare dei veri e propri router basati su Linux.

I dati che transitano in una rete sono divisi in pacchetti di dimensioni prefissate, con netfilter è possibile controllare il contenuto di ogni singolo pacchetto, e definire le azioni da compiere in base alle caratteristiche di quelli ricevuti. Ad esempio, si può definire una regola che impedisce la ricezione di pacchetti provenienti da un particolare indirizzo o che utilizzano una determinata porta per effettuare la connessione.

Il sistema netfilter è basato su regole raggruppate in *catene* (*chain*), a loro volta raggruppate in *tabelle* (*tables*). Ogni tabella definisce un tipo diverso di operazioni che è possibile effettuare sui pacchetti; ogni catena definisce come vengono trattati i pacchetti nelle diverse fasi della loro elaborazione.

Le catene sono una forma di lista di controllo degli accessi: ogni regola è costituita da due parti: la specifica delle caratteristiche che un pacchetto deve avere affinché la regola stessa venga applicata (*match*) e un *obiettivo* o *target*, che indica cosa fare quando il pacchetto rispetta le caratteristiche indicate. A ciascuna catena è anche associata una *politica* di default, che definisce come vengono trattati i pacchetti che non corrispondono ad alcuna regola.

Le caratteristiche più di frequente utilizzate per costruire delle regole sono l'indirizzo di partenza o di destinazione del pacchetto e il numero di porta associato alla connessione. Ogni pacchetto di rete che arriva o parte dal computer attraversa almeno una catena e ogni regola della catena controlla se il pacchetto ne rispetta la specifica. Se questo accade, il pacchetto seguirà il comportamento descritto nell'obiettivo della regola, e le regole successive della catena verranno ignorate (a parte casi speciali). Se il pacchetto raggiunge la fine della catena senza essere processato da nessuna regola, la politica della catena determina cosa farne.

Uno dei possibili obiettivi è il collegamento a un'altra catena. In questo caso, il pacchetto ricomincia ad essere valutato dalle regole della nuova catena, senza limiti di concatenazione. Una regola può semplicemente essere un collegamento a una catena. Solo se il pacchetto attraversa l'intera catena collegata esso continuerà nella catena principale.

In ogni tabella esistono alcune catene predefinite, ma l'utente può crearne di nuove.

Esistono tre tabelle prestabilite, ognuna delle quali contiene delle catene predefinite. Esiste anche la possibilità di creare altre tabelle. L'amministratore può creare e cancellare le catene definite dall'utente in qualsiasi tabella. Inizialmente, tutte le catene sono vuote e hanno una politica che permette a tutti i pacchetti di passare senza essere bloccati o alterati in alcun modo, esse vanno poi modificate a seconda delle proprie esigenze.

Le tabelle predefinite sono le seguenti:

- Tabella filtro (filter): è responsabile del filtraggio dei pacchetti, permette cioè di bloccarli o di farli passare. Ogni pacchetto passa attraverso la tabella filtro. Essa contiene le seguenti catene predefinite:
    - Catena INPUT: tutti i pacchetti destinati al sistema passano attraverso questa catena.
    - Catena OUTPUT: tutti i pacchetti creati dal sistema passano attraverso questa catena.
    - Catena FORWARD: tutti i pacchetti che hanno come destinazione finale un altro sistema e che non sono stati generati dal sistema stesso, cioè tutti i pacchetti che vengono meramente instradati dal sistema, passano attraverso questa catena.

- tabella nat:

  Questa tabella è responsabile dell'impostazione delle regole per la modifica degli indirizzi e porte dei pacchetti. Il primo pacchetto di una connessione passa attraverso questa tabella, e il risultato del passaggio del primo pacchetto determina come tutti gli altri pacchetti della stessa connessione verranno modificati. La tabella nat contiene le seguenti catene predefinite:

    - Catena PREROUTING: passano attraverso questa catena i pacchetti in entrata, il passaggio avviene prima che la locale tabella di routing venga consultata per effettuare l'instradamento. Essa è usata per il NAT sulla destinazione o DNAT.
    - Catena POSTROUTING: passano attraverso questa catena i pacchetti in uscita dopo che la locale tabella di routing sia stata consultata. Usata per il NAT sulla sorgente o SNAT.
    - Catena OUTPUT: permette un DNAT limitato sui pacchetti generati localmente.

- tabella mangle:

  questa tabella è responsabile delle modifiche alle opzioni dei pacchetti, come ad esempio quella che determina la qualità del servizio. Tutti i pacchetti passano attraverso questa tabella.

Essa contiene tutte le catene predefinite:

- Catena PREROUTING: esamina tutti i pacchetti che in qualche modo entrano nel sistema. Questo processo avviene prima che il routing decida se il pacchetto debba essere inoltrato (catena FORWARD) o se sia destinato al sistema. Viene utilizzata per manipolare l'header del pacchetto (catena INPUT).
- Catena INPUT: tutti i pacchetti destinati al sistema passano per questa catena.
- Catena FORWARD: tutti i pacchetti che vengono instradati dal sistema ma di cui il sistema non è ne sorgente iniziale ne destinazione finale, passano per questa catena.
- Catena OUTPUT: tutti i pacchetti generati dal sistema passano per questa catena.
- Catena POSTROUTING: tutti i pacchetti che lasciano il sistema, sia quelli in OUTPUT sia quelli in FORWARD, passano poi per questa catena.

L'obiettivo di una regola è l'azione da compiere se un pacchetto rispetta la regola, e viene specificato con la seguente opzione:

`-j` *`obiettivo`*

`--jump` *`obiettivo`*

L'obiettivo può essere:

- Una catena definita dall'utente
- uno degli obiettivi predefiniti (`ACCEPT`, `DROP`, `QUEUE`, o `RETURN`)
- Uno degli obiettivi aggiuntivi, come ad esempio REJECT o LOG.
- Se una regola non contiene la specifica dell'obiettivo, il destino del pacchetto non verrà modificato, tuttavia il contatore della regola verrà ugualmente incrementato.

Quando l'obiettivo è il nome di una catena definita dall'utente, il pacchetto viene fatto passare per quella catena. Se il pacchetto non viene processato da nessuna regola della catena, in quanto non rispetta la specifica di nessuna delle sue regole, esso ritorna ad essere processato dalla catena di partenza.

Ogni obiettivo predefinito indica un'azione da compiere sul pacchetto:

`ACCEPT` — accetta

Questo obiettivo comporta che netfilter accetterà il pacchetto. Il risultato pratico di questa accettazione dipende da quale catena sta processando il pacchetto. Per esempio, un pacchetto che è accettato dalla catena INPUT può essere ricevuto dal sistema, un pacchetto accettato dalla catena OUTPUT può essere inoltrato dal sistema, e un pacchetto accettato dalla catena FORWARD potrà essere smistato dal sistema a un'altra destinazione, un pacchetto "accettato" in una catena della tabella <t>nat</t> non subirà alterazioni.

`DROP` — scarta

Questo obiettivo determina che il pacchetto venga scartato senza effettuare ulteriori operazioni su di esso. Il pacchetto scomparirà senza che alcuna indicazione del fatto che sia stato scartato venga fornita all'applicazione o al sistema che ha inviato il pacchetto. Il mittente del pacchetto vedrà semplicemente scadere il tempo a disposizione per la comunicazione, e non potrà distinguere tra il caso in cui il pacchetto è stato ricevuto e poi scartato e il caso in cui il pacchetto non è mai stato ricevuto. Questo comportamento aumenta la sicurezza di un sistema in quanto un potenziale nemico non potrà neppure determinare se il sistema esiste effettivamente.

`QUEUE` — metti in coda

Questo obiettivo fa in modo che il pacchetto sia inserito in una coda, in modo che possa essere processato da una applicazione. La libreria libipq, facente parte del progetto netfilter/iptables, permette a un’applicazione di modificare i pacchetti inseriti in una coda. Se non vi è nessuna applicazione che processa i messaggi in coda, questo obiettivo sarà equivalente all'obiettivo DROP.

`RETURN` — ritorna

Questo obiettivo ha lo stesso effetto di raggiungere la fine della catena: per una regola nella catena predefinita, viene eseguita la politica della catena; per una regola definita dall'utente, l'attraversamento delle regole continua nella catena chiamante, subito dopo il punto in cui è presente la catena che ha causato il RETURN, analogamente a come accade nella chiamate a funzione.

Esistono molti obiettivi aggiuntivi disponibili. Alcuni dei più comuni sono:

`REJECT` — rifiuta

Questo obiettivo ha lo stesso effetto di DROP con l'eccezione che viene spedito un pacchetto di errore ICMP al mittente del pacchetto. Esso è principalmente utilizzato nelle catene INPUT o FORWARD della tabella filtro. Un pacchetto di errore può indicare esplicitamente che il pacchetto è stato filtrato.

`LOG` — annota

Con questo obiettivo il pacchetto viene annotato, cioè la ricezione del pacchetto viene annotata inviando un messaggio sul SysLog. Questo obiettivo può essere utile per permettere all'amministratore di sapere quali pacchetti vengono filtrati o allo sviluppatore per controllare il corretto funzionamento del sistema.

`DNAT(Destination nat)`

Questo obiettivo comporta la riscrittura dell'indirizzo di destinazione del pacchetto, per permettere il NAT sulla destinazione. Questo obiettivo è valido esclusivamente nelle catene OUTPUT e PREROUTING della tabella nat. La decisione effettuata sul primo pacchetto verrà ripetuta per tutti i pacchetti della connessione, e i pacchetti di risposta avranno l'indirizzo sorgente originario.

`SNAT(Source nat)`

Questo obiettivo comporta la riscrittura dell'indirizzo del mittente del pacchetto, per permettere il NAT sulla sorgente. Questo obiettivo è valido solo nella catena POSTROUTING della tabella nat, e come DNAT il suo risultato è ripetuto per tutti i pacchetti della stessa connessione.

`MASQUERADE` — maschera

Questa è una forma speciale di SNAT per indirizzi IP dinamici, come quelli forniti da molti Internet Service Provider per i loro utenti.

Iptables è un'applicazione che permette agli amministratori di configurare le tabelle, le catene e le regole di netfilter. Dato che iptables modifica il funzionamento del sistema operativo, per essere eseguito è necessario entrare nel sistema come utente amministratore, che nei sistemi di tipo Unix è

l'utente root, il quale ha i permessi per compiere qualsiasi tipo di operazione. Sulla maggior parte dei sistemi Linux, iptables è installato come `/usr/sbin/iptables`. La lista completa delle funzionalità del comando è consultabile nella relativa documentazione, che può essere visualizzata con il comando "`man iptables`".

Firewall Builder

Firewall Builder è un'interfaccia grafica per la configurazione e la gestione di una firewall che supporta iptables netfilters oltre a molti altri.

Con Firewall Builder un amministratore può gestire più firewall usando lo stesso database. La modifica di un oggetto si riflette immediatamente sulla gestione di quell'oggetto da parte di questi ultimi.

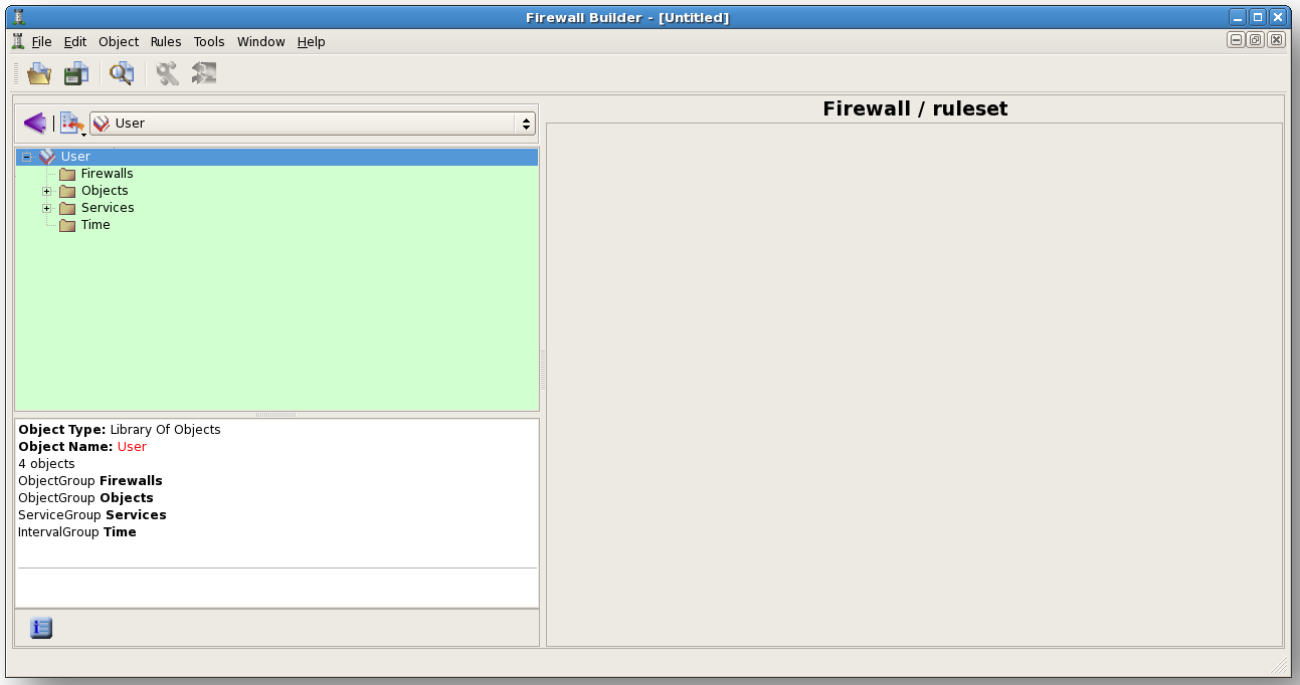
L'installer interattivo Built-in usa l'SSH per comunicare con il firewall e può automaticamente copiare la politica generata ed attivarla. Supporta la modalità batch e può aggiornare la politica dei diversi firewall in una sola sessione.

In questa applicazione l'amministratore lavora con un'astrazione della politica del firewall e le regole NAT; il software nasconde le specifiche di un particolare target e aiuta l'amministratore a concentrarsi sull'implementazione della politica di sicurezza.

La maggior parte delle distribuzioni linux comprende all'interno del sistema il pacchetto d'installazione di Firewall Builder. Spesso è già installato e il collegamento per l'avvio veloce è già disponibile. Tuttavia se così non fosse si può sempre avviare digitando da riga di comando:

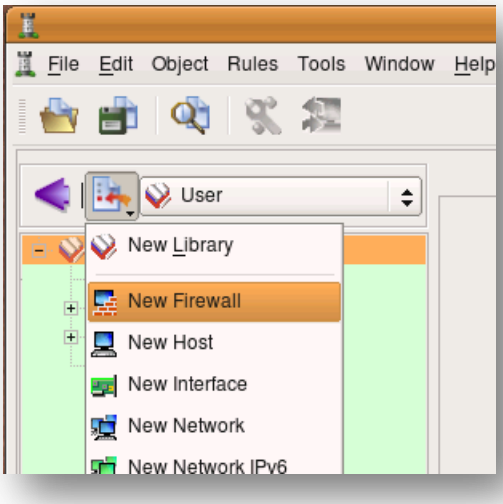
```
fwbuilder
```

Il programma si avvierà e si aprirà la finestra principale:



La parte sinistra della prima schermata rappresenta tramite un diagramma ad albero, la lista degli oggetti.

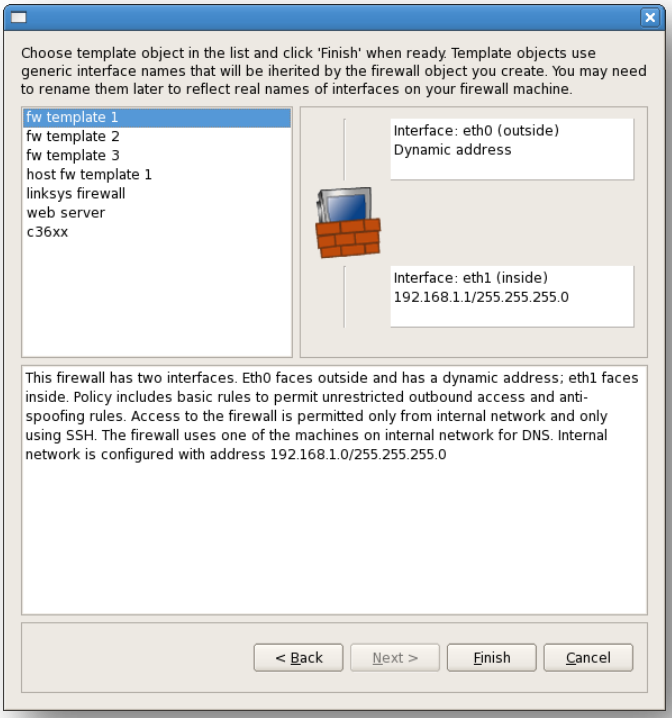
A questo punto si può passare alla creazione del firewall, che effettivamente non la realizzazione vera e propria di un firewall, ma soltanto la creazione di un file .fwb, che conterrà la configurazione completa del firewall.



Nella schermata successiva verrà chiesto all'utente di selezionare quale firewall utilizzare, e al quale FwBuilder dovrà fare da interfaccia grafica. Nel caso in questione verrà selezionato IPTables.

Viene anche chiesto di specificare la versione del sistema operativo installato.

Apponendo il segno di spunta nell'apposita casella si sceglierà di utilizzare un template preconfigurato. In questa maniera si può selezionare il modello della rete su cui si sta lavorando, dando al programma le informazioni necessarie per poi poter configurare al meglio il firewall.

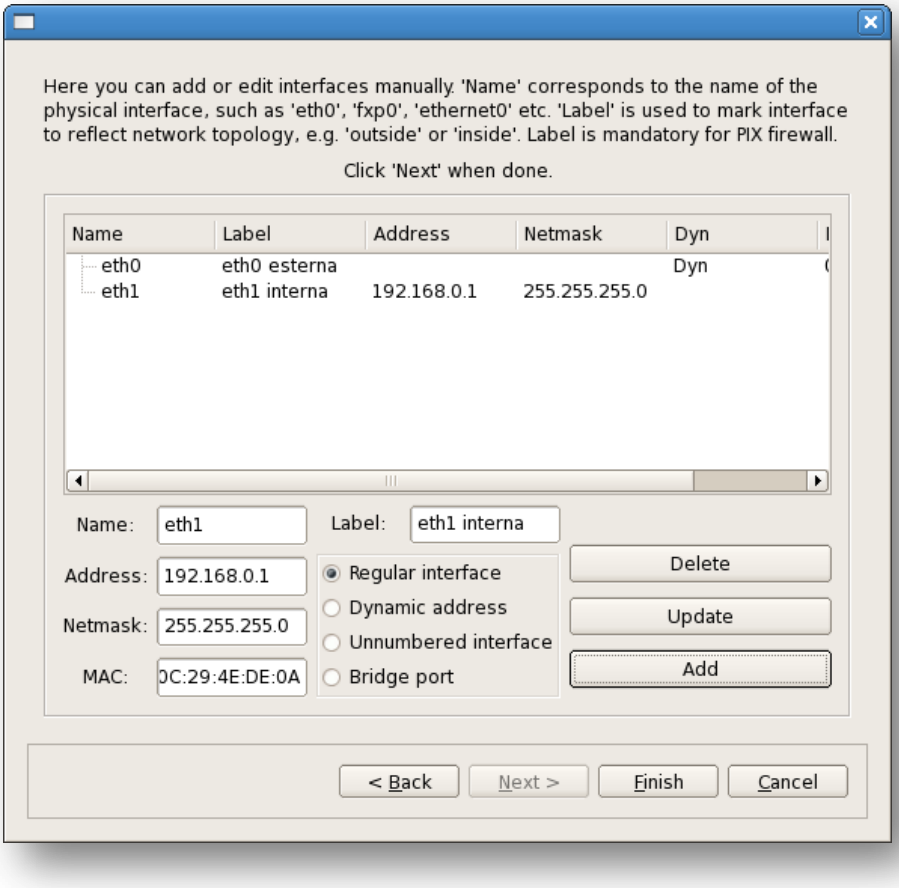


Il tipo di configurazione selezionato presenta un sistema che ha due interfacce, e cioè due schede di rete, esattamente come il sistema precedentemente realizzato.

L'interfaccia Eth0 è rivolta verso l'esterno, quindi verso Internet, avrà un indirizzo IP dinamico, mentre la periferica Eth1, rivolta verso la sottorete interna, sarà fornita di un indirizzo IP, assegnatogli dal DHCP. L'accesso al firewall viene permesso soltanto se proveniente dall'interno della sottorete e esclusivamente via SSH.

La rete interna sarà configurata con indirizzi 192.168.1.0/255.255.255.0

Al contrario se viene deselezionata la casella nella schermata di cui sopra, sarà possibile configurare manualmente il tipo di struttura della rete di cui si ha bisogno e su cui dovrà lavorare il Firewall.



In questa finestra si possono vedere diversi campi da personalizzare.

Avendo la necessità di configurare due schede di rete, si procederà con ordine iniziando dalla prima, e cioè quella rivolta verso la Rete esterna.

All'interno del campo "Name" va quindi inserito il nome della periferica, in questo caso Eth0.

Essendo rivolta verso Internet è necessario assegnare un indirizzo di tipo dinamico, e per farlo è sufficiente spuntare la casella di fianco.

La Netmask sarà 255.255.255.0, mentre il MAC address dovrà essere quello relativo a tale dispositivo di rete. Una volta fatto ciò, e dopo aver inserito un nome generico nel campo "Label",

(nella schermata è “eth1 interna”) basterà aggiungere la configurazione appena preparata mediante il pulsante “Add”.
La seconda scheda di rete a differenza della prima, sarà posizionata verso la sottorete interna, e quindi non ci sarà bisogno di utilizzare l’indirizzo IP dinamico, e sarà sufficiente assegnarne uno del tipo 192.168.0.1 come in questo caso.

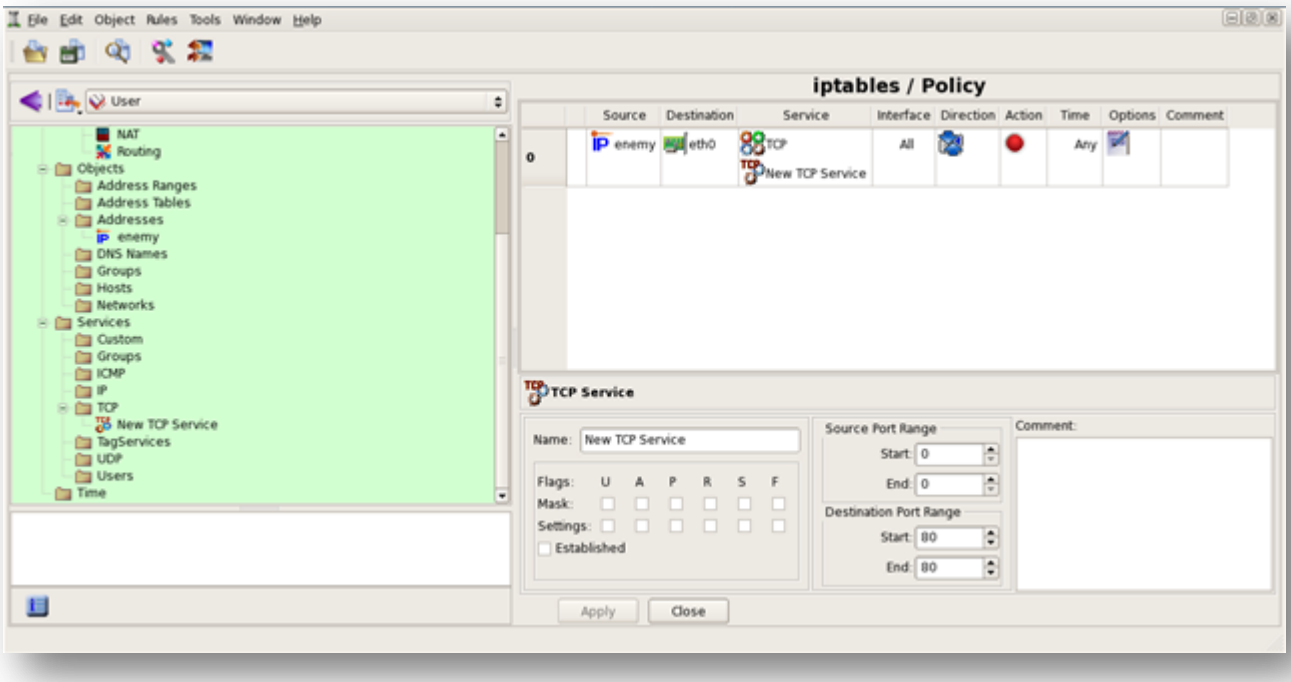
Una volta riempiti tutti i campi e aggiunta anche la seconda scheda il firewall verrà creato.

Una volta fatto ciò si passa alla definizione delle regole vere e proprie.

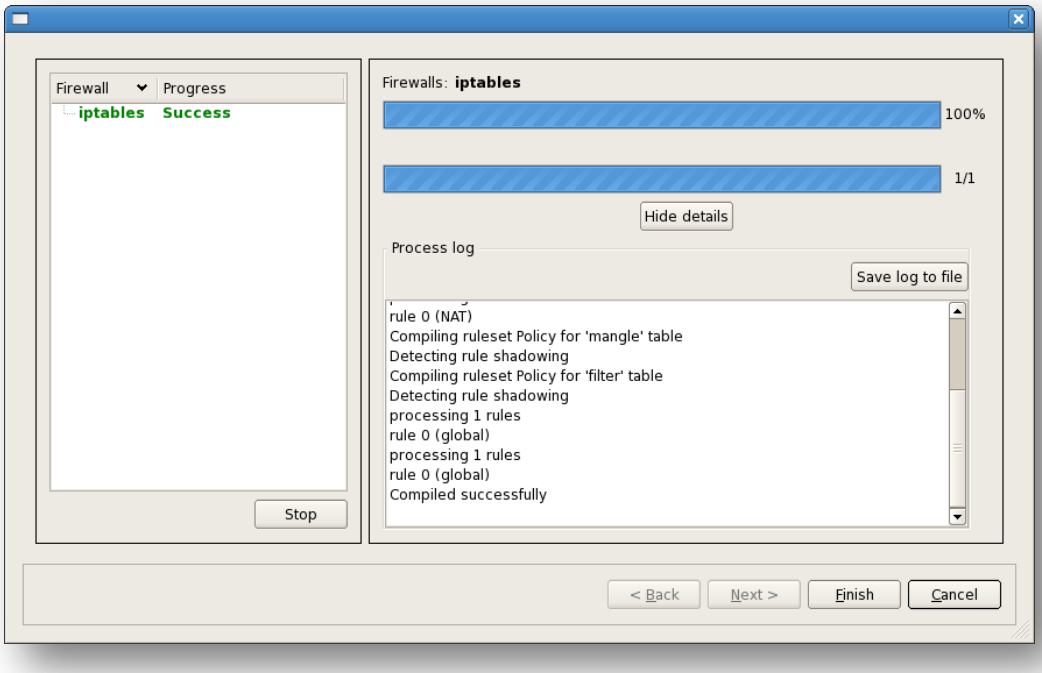
Attraverso il menu a tendina “Rules”, e selezionando la voce “Insert Rule”, è possibile creare da zero la propria regola.

In questo caso è stata definita una regola che indica al firewall di non accettare pacchetti di tipo TCP dall’indirizzo specificato, e provenienti dalla scheda di rete rivolta verso Internet. Quindi questi pacchetti verranno controllati, e se avranno nell’header avranno come indirizzo di provenienza quello indicato verranno scartati.. Questo indirizzo viene identificato come “Enemy”.

Ovviamente l’azione che il firewall dovrà compiere sarà negare l’accesso al traffico proveniente da Enemy, come si evince dallo screenshot seguente.



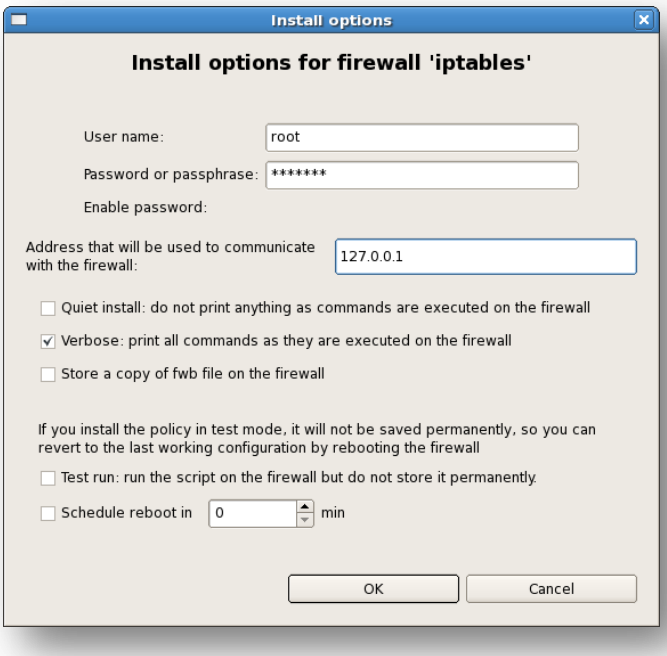
Cliccando il tasto “compila” il programma crea uno script che se avviato applica le regole precedentemente create da IPTables.



L’installazione non fa altro che eseguire lo script creato durante la compilazione e quindi le regole descritte nella configurazione vengono rese attive, sul Firewall.

Durante le fasi d’installazione verranno richieste all’utente lo username e la password per poter installare le regole su una qualsiasi macchina definita dall’utente stesso.

Per effettuare l’installazione è necessario avere i permessi di root.



IDS/IPS

IDS, Intrusion Detection System é un dispositivo software o hardware che viene utilizzato per identificare e intercettare accessi non autorizzati a un computer o a una rete di computer. A volte vengono utilizzate combinazioni di dispositivi hardware e software già preinstallati e configurati.

Le intrusioni possono essere effettuate da cracker esperti o da semplici utenti della rete muniti di tools semiautomatici. Gli attacchi possono sfruttare servizi vulnerabili, o essere inviati sottoforma di pacchetti modificati e malformati o applicazioni malevole. Tentativi di accesso agli host possono essere compiuti attraverso l'innalzamento illecito dei privilegi degli utenti, che danno la possibilità di accedere a computer e file a utenti non autorizzati.

Altri attacchi sono i più comuni virus, trojan e worm.

Un IDS è composto da quattro componenti. Uno o più *sensori* utilizzati per ricevere le informazioni dalla rete o dai computer. Una *console* utilizzata per monitorare lo stato della rete e dei computer e un *motore* che analizza i dati prelevati dai sensori e provvede a individuare eventuali falle nella sicurezza informatica. Il motore di analisi si appoggia a un *database* ove sono memorizzate una serie di regole utilizzate per identificare violazioni della sicurezza. Esistono diverse tipologie di IDS che si differenziano a seconda del loro compito specifico e delle metodologie utilizzate per individuare violazioni della sicurezza. Il più semplice IDS è un dispositivo che integra tutte le componenti in un solo apparato.

Un IDS consiste quindi in un insieme di tecniche e metodologie realizzate ad-hoc per rilevare pacchetti sospetti a livello di rete, di trasporto o di applicazione. Due sono le categorie base: sistemi basati sulle firme (signature) e sistemi basati sulle anomalie (anomaly). La tecnica basata sulle firme è in qualche modo analoga a quella per il rilevamento dei virus, che permette di bloccare file infetti e si tratta della tecnica più utilizzata. I sistemi basati sul rilevamento delle anomalie utilizzano un insieme di regole che permettono di distinguere ciò che è "normale" da ciò che è "anormale". È importante sapere che un IDS non può bloccare o filtrare i pacchetti in ingresso e in uscita, né tanto meno può modificarli. Un IDS può essere paragonato a un antifurto e un firewall a una porta blindata. L'IDS non cerca di bloccare le eventuali intrusioni, cosa che spetta al firewall ma cerca di rilevarle laddove si verifichino.

I meccanismi d'individuazione di attività sospette sono diversi, ma generalmente si concentrano su:

- verifica dei log di sistema o di specifici programmi per individuare attività anomale;
- controllo dell'integrità dei file locali (modifiche sospette possono essere sintomo di una avvenuta irruzione);
- monitoring dei pacchetti destinati all'host, sia per reagire a pattern di attacco noti che per accorgersi di un port scan remoto, generalmente prologo di un tentativo di intrusione.

Gli IDS si suddividono in due ulteriori categorie, gli IDS passivi e degli IDS attivi. I primi IDS quando rilevano una violazione della sicurezza informatica provvedono a notificarla all'operatore tramite la console ed eventualmente gli inviano una email. Gli IDS attivi oltre a notificare all'operatore una violazione della sicurezza provvedono a prendere delle opportune contromisure per eliminare o comunque isolare la violazione informatica.

Nei sistemi attivi l'eliminazione della violazione si ottiene usualmente riprogrammando la lista di controllo degli accessi del firewall in modo da impedire l'accesso agli indirizzi responsabili dell'attacco. Questa tipologia di IDS va accuratamente programmata dato che una falsa identificazione potrebbe bloccare un utente autorizzato.

Il firewall non è in grado di bloccare violazioni della sicurezza che avvengono dall'interno della rete locale. A questo scopo sono stati sviluppati gli Intrusion prevention system. Questi componenti contengono delle liste programmate dall'IDS che vengono utilizzate per decidere se un programma deve essere mandato in esecuzione o no. Questi componenti impediscono a worms o virus di diffondersi nei vari computer dato che il componente ne impedisce l'attivazione.

Gli Intrusion prevention system sono dei componenti sviluppati per incrementare la sicurezza informatica. Sono stati sviluppati per impedire ad un programma non autorizzato di entrare in esecuzione. La tecnologia "Intrusion prevention" spesso viene considerata come un'estensione della tecnologia intrusion detection (IDS) sebbene sia più simile ad una lista di controllo degli accessi di un firewall. Evita l'attivazione di programmi potenzialmente malevoli. Questi sistemi hanno un numero molto basso di falsi positivi e possono essere utilizzati in congiunzione con gli IDS per evitare la propagazione di virus o worm.

Gli IDS possono anche sfruttare database, librerie e firme di attacco (o signature) per rilevare le intrusioni. Quando il traffico di rete oppure un'attività di rete corrisponde a una regola ben nota all'ids, questi segnala il tentativo di intrusione. Il limite principale è che l'affidabilità di tale strumento dipende interamente dalla tempestività con cui il database degli attacchi viene aggiornato. Gli Ids basati sulle regole funzionano in due modalità: una preventiva e una reattiva e sono due modalità che cambiano la tempistica di azione e la possibilità di interazione. Il primo approccio, di tipo reattivo, permette di completare alla perfezione la procedura di logging: il sistema avverte che si è verificato un attacco, anche se è trascorso qualche minuto dall'evento, provvedendo a notificarlo all'operatore tramite la console o inviando una e-mail. Diversamente l'approccio preventivo risponde in tempo reale all'attacco in corso consentendo di rintracciare la sua origine. Oltre ad avvisare all'amministratore la violazione, è in grado di prendere delle contromisure per eliminare, o comunque isolare, la violazione. Questi due metodi hanno il grande problema di generare falsi positivi (attività anomale che non sono intrusive, ma che vengono segnalate come tali) e falsi negativi (tutte le attività che sono anomale e che non vengono rilevate e segnalate). L'uso di un solo metodo non può offrire una totale sicurezza; la situazione più grave si presenta nel caso dei falsi negativi, poiché può compromettere gravemente la sicurezza del sistema, ma anche un'eccessiva presenza di falsi positivi può portare a un'impossibilità di utilizzo del computer per un eccesso di avvisi di intrusione infondati.

Snort

L’ IDS preso in considerazione è Snort.

Snort è un software open source per la prevenzione delle intrusioni di rete, con un sistema di rilevazione che utilizza delle regole per effettuare tali rilevazioni. Snort è la tecnologia di rilevamento delle intrusioni e prevenzione più diffusa al mondo, che con milioni di download fino ad oggi, è diventata lo standard dell’industria.

Snort é stato progettato per funzionare in tre diversi modi, anche se poi più avanti verrà illustrata una quarta possibilità di funzionamento, l’InLine Mode.

Comunque le modalità di base sono:

- Sniffer
- Packet Logger
- Network Intrusion Detection (meglio conosciuta come NIDS, dove la S sta per system.)

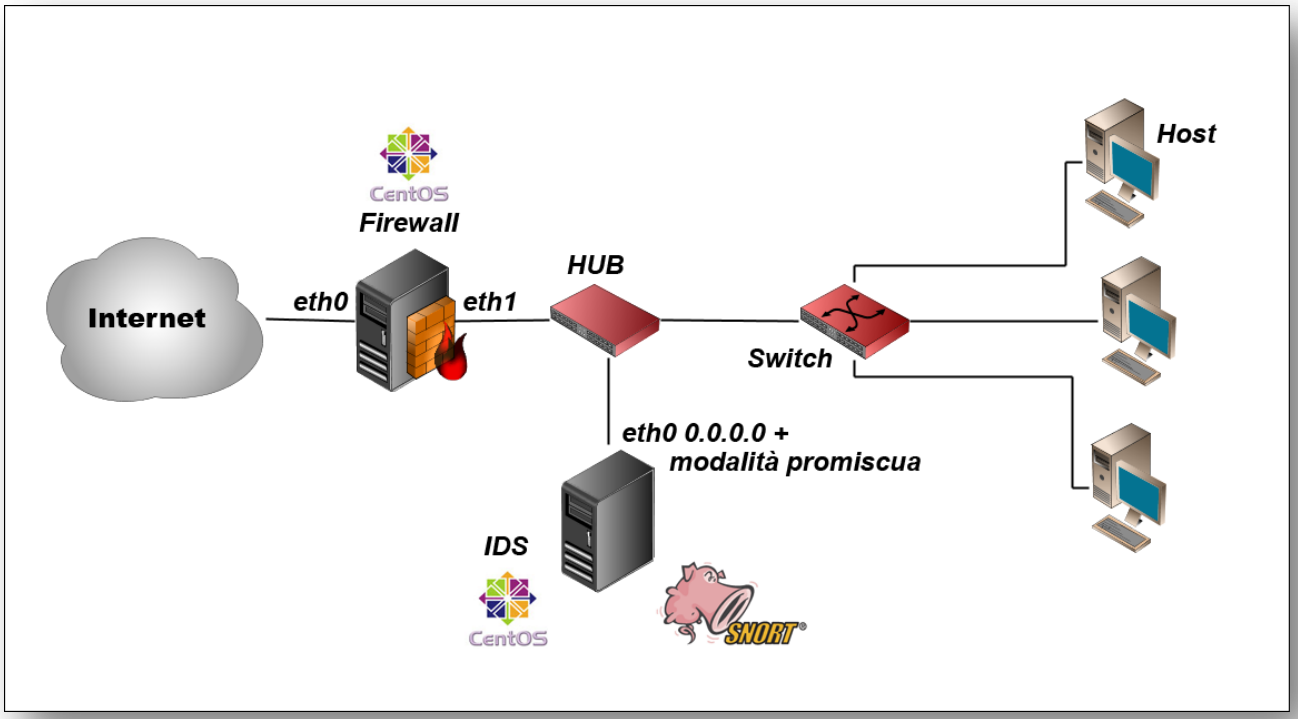
La modalità Sniffer server a intercettare i pacchetti che viaggiano sulla rete.

Il Packet logger copia sul disco locale una copia dei pacchetti in transito.

L'ultima modalità è il NIDS, esso analizza il traffico dei pacchetti attraverso la rete e mediante una serie di regole configurabili ed esegue su di essi della operazioni in base al loro contenuto.

Va specificato però che Snort nelle tre modalità sopra descritte NON blocca il traffico dei pacchetti, ma si limita soltanto ad analizzarne il contenuto.

La configurazione migliore perché Snort funzioni al meglio sarebbe questa:



Il sistema prevede che esista una macchina con due schede di rete, rispettivamente Eth0 e Eth1, sulla quale viene installato un firewall. La periferica Eth0 sarà rivolta verso l'esterno e quindi esposta al traffico Internet. La seconda scheda invece sarà collegata verso l'interno attraverso un HUB e uno Switch a una rete interna, con un indirizzo fisso del tipo 192.169.69.1. La macchina in questione provvederà a fornire l'intera sottorete di indirizzi IP,tramite DHCP, fornendo anche i servizi di DNS e Dyn-DNS. Collegato all'HUB ci sarà un computer, sul quale verrà installato Snort. La macchina sarà collegata alla sottorete in modalità promiscua, cioè senza che le venga assegnato un IP, rendendosi completamente invisibile a chiunque. In questo modo Snort sarebbe libero di "sniffare" tutto il traffico in transito verso e dalla sottorete, in modo da poter fermare più facilmente gli attacchi e non essere a sua volta bersaglio di cracker e simili.

Installazione

Prima di iniziare a installare Snort sulla propria macchina, occorrerà caricare altri programmi.

Nel caso in esame, poiché si è scelto di utilizzare Snort attraverso interfacce grafiche e vi è anche l'utilizzo di un database, bisognerà procurarsi i seguenti pacchetti:

- mysql
- mysql-bench
- mysql-server
- mysql-devel
- php-mysql
- httpd
- pcre-devel
- php
- php-gd
- mod-ssl
- gd
- libpcap-devel
- gcc
- gcc-c++
- glib2-devel

Alcuni servizi, con l'utilizzo di Snort si renderanno necessari all'avvio del sistema.

Per ottenere ciò basta fare:

```
Chkconfig httpd on
Chkconfig mysql on
Service httpd start
Service mysqld start
```

È possibile reperire Snort sul sito ufficiale, all'indirizzo http://www.snort.org/.

Nella sezione "Get Snort" è possibile trovare i pacchetti .tar.gz contenenti i codici sorgenti del programma.

Altrimenti è possibile scaricare direttamente l'applicazione in formato precompilato.

Prima di procedere con l'installazione di Snort è necessario verificare la presenza nel sistema della librerie "libpcap", che serviranno alla "cattura" dei pacchetti. Inoltre è fondamentale la presenza di un database e quindi di un sistema per la gestione di quest'ultimo.

MySQL

Per far sí che Snort abbia la possibilità di loggare gli alert è richiesta la presenza di una database, sul quale tenere traccia degli avvenimenti verificatisi.

Nel caso specifico sarà adottato un database system management open source, MYSQL.

MYSQL è un DBMS composto da un client e un server, entrambi disponibili sia per Linux che per piattaforme Windows.

L'installazione di MySQL comprende generalmente l'uso di diversi pacchetti (o compilazione delle relative componenti):

`mysql-server` - Il server vero e proprio

`mysql-client` - Il client a riga di comando

`mysql-devel` - Librerie. Da installare solo se si devono compilare software che si appoggiano a MySQL

Il file di configurazione è `/etc/my.cnf`.

I comandi fondamentali sono:

`mysqld` - Il server MySQL
`mysql` - Il client a riga di comando. Per usarlo bisogna conoscere i principi di SQL.
`mysqladmin` - Tool di gestione e amministrazione del sistema
`mysqldump` - `mysqlhotcopy` - Strumenti per il backup di un database

La gestione dei permessi sui database di MySQL è fondamentale.
Di default esiste l'utente root senza password che da localhost può gestire completamente MySQL. Notare che questo utente NON è l'utente root di sistema, le sue credenziali, come quelle di tutti gli altri utenti di MySQL, sono presente nel database predefinito mysql.
In termini di gestione delle password e degli accessi si consiglia:
- Impostare una password per l'utente root.
- Impostare login/password diverse per ogni applicativo o sito web che lavora su un singolo database, in modo che questo utente possa lavorare solo su quello specifico db.
- Se da applicativo o interfaccia web si devono solo interrogare dati, bastano i permessi di SELECT, se si devono poterli modificare, dovrebbero bastare permessi di SELECT, UPDATE, INSERT, DELETE. Gli altri permessi servono per modificare la struttura di un database, impostare indici, impostare permessi ecc. e generalmente non è necessario attivarli, salvo quando l'applicazione lo richieda esplicitamente.

- La porta su cui ascolta MySQL (TCP 3306) non dovrebbe mai essere esposta ad Internet, è sufficiente che sia accessibile dall'host su cui gira l'applicativo che usa il DB server (esempio un server web con pagine php).

Per configurare MYSQL in maniera adeguata affinché collabori con Snort c'è bisogno di effettuare alcune modifiche.

Siccome all'inizio MYSQL ha un solo utente root, privo di password c'è bisogno di effettuare la creazione di almeno due di essi.

Per prima cosa si procederà alla definizione dell'utente "root".

All'interno di un terminale si inserirà il comando:

```
mysql
```

Per entrare nel client di MYSQL.

E di seguito:

SET PASSWORD For root@localhost=PASSWORD('centosql');

In questo modo é stato definito l'utente root sulla macchina su cui si sta lavorando(loclahost) con password "centosql".

Ora si passa alla creazione del database che verrà utilizzato da Snort:

create database snort;

per concedere i permessi allo user "root" di scrivere e ricercare sul database si utilizzerà il comando:

grant INSERT,SELECT on root.* to snort@localhost.

Ora si ha l'utente root con relative password, non resta quindi che definirne un secondo, che in questo caso verrà chiamato "snort"

SET PASSWORD for snort@localhost = PASSWORD('snortdb');

grant CREATE,INSERT,SELECT,DELETE,UPDATE on snort.* to snort@localhost;

exit

Con questi comandi viene creato un nuovo utente "snort" con permessi di creare, inserire dati, selezionare, eliminare e aggiornare il database snort su tutte le tabelle che lo compongono (.*), il tutto effettuato sulla macchina locale.

A questo punto si avrà un database del tutto vuoto.

Attraverso uno script presente nell'installazione di Snort, "create_mysql, si andranno a definire le tabelle presenti all'interno del database:

```
mysql -u root -p < /usr/share/snort-x.x.x/schemas/create_mysql snort
password centosql
```

Per verificare che tutto sia andato a buon fine:

```
mysql -u root -p
```

Una volta reperiti i file dal sito si può procedere con l'installazione del programma.

Se si è scelto di prendere i pacchetti rpm, cioè quelli precompilati sarà sufficiente aprire una Shell e scrivere:

```
rpm -i snort-x.x.x
```

I pacchetti rpm sono già precompilati quindi non sarà necessario compilarli. In questo modo installare un programma è molto più semplice e pratico, anche se per questa occasione è stato scelto di compilare a mano ogni programma installato per motivi di maggiore affidabilità delle applicazioni, in quanto era strettamente necessario avere un sistema quanto più stabile.

Di seguito verrà mostrata quindi anche l'installazione da pacchetti non precompilati.

Una volta scaricato il file .tar.gz relativo alla versione di Snort che si vuole utilizzare, bisognerà estrarre il contenuto in una cartella a piacere che per praticità sarà chiamata anche'essa Snort.

Ora si mostra come apparirà l'inserzione dei comandi per l'installazione, per semplicità vengono riportate soltanto le prime righe dell'output che il programma produrrà, in quanto risulta inutile mostrarlo nella sua interezza.

Attraverso il commando ./configure si fa in modo di verificare se sul sistema ci sia il necessario per il corretto funzionamento, mentre con -with mysql si controlla che sia presente e funzionante MYSQL e che sia predisposto alla collaborazione con Snort.

```
[root@root snort-1.9.0]# ./configure –with mysql
loading cache ./config.cache
checking for a BSD compatible install... (cached)
/usr/bin/install -c
checking whether build environment is sane... yes
checking whether make sets ${MAKE}... (cached) yes
........
```

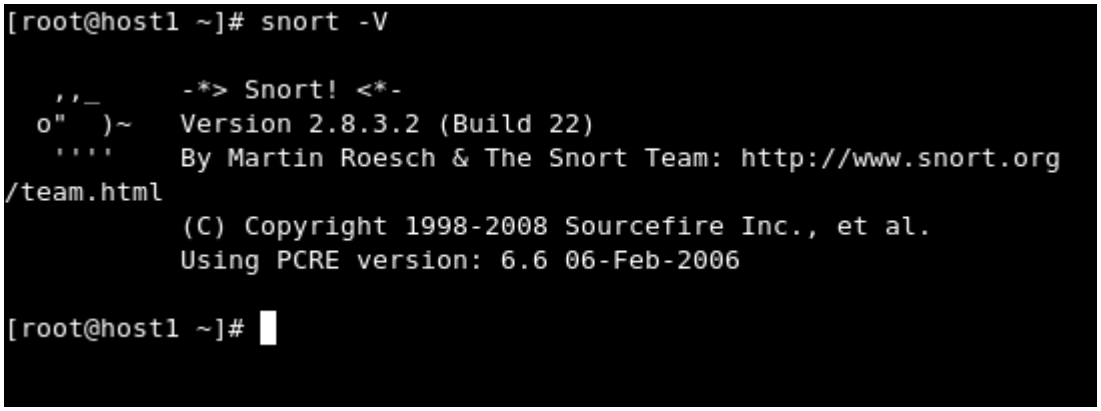
```
root@root snort-1.9.0]# make
cd . && /root/snort/snort-1.9.0/missing autoheader
.....

[root@root snort-1.9.0]# make install
Making install in src
make[1]: Entering directory `/root/snort/snort-1.9.0/src'
```

Snort è ora installato sul sistema in `/usr/local/bin/snort`.

Per verificare che tutto si andato a buon fine digitare:

```
snort –V
```

Se la schermata sarà simile a questa, vuol dire che Snort ora è presente sul sistema.



Ora si rende necessario creare dei gruppi di utenti, ai quali sarà poi possibile assegnare dei permessi.

Ovviamente assegnare un permesso a un gruppo significherà concederli a ogni membro.

```
/usr/sbin/groupadd snort
```

```
/usr/sbin/useradd –g snort snort –s /sbin/nologin
```

Ora si devono creare le directory nelle quali andranno poi i file di configurazione, le regole e i log che Snort produrrà in base al traffico analizzato.

`mkdir /etc/snort` in questa cartella verrà creato da Snort il file di configurazione .conf che verrà analizzato in seguito più nel dettaglio.

`mkdir /etc/snort/rules` è l'ubicazione delle regole che il programma userà per riconoscere i diversi tipi di attacchi ricevuti.

Sul sito di riferimento sono presenti due tipi diversi di regole, una per gli utenti normali, mentre l'altra, più completa necessita di una registrazione per essere scaricata.

Per la realizzazione di questo sistema sicuro sono state scaricate entrambe le versioni, per poter effettuare un maggior numero di prove con molteplici modalità di attacco.

`mkdir /var/log/snort` é dove verranno depositati file di log prodotti da Snort in seguito ad attacchi rilevati.

Questi comandi, se si dovesse utilizzare il pacchetto rpm, non saranno necessari, poiché il lavoro è già svolto.

All'interno del pacchetto .tar.gz scaricato in precedenza è presente una cartella denominata "etc", essa va spostata nella directory /etc/snort.

Arrivati a questo punto è possibile procedere con la modifica, cioè la customizzazione del file

snort.conf .

In particolare vanno presi in considerazione dei parametri, e le impostazioni relative al sistema che si sta realizzando sono:

```
var HOME_NET 192.168.69.0/24

var EXTERNAL_NET !$HOME_net
```

Con questo commando si identifica tutto ció che non rappresenta la rete interna e quindi avendo due schede di rete ci si riferisce al traffico proveniente dall'esterno.

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

Identica la cartella in cui vanno inserite le regole

```
output database: log,mysql,user=snort password = snortdb dbname = snort
host = localhost

output database: alert,mysql,user=snort password = snortdb dbname = snort
host = localhost
```

Interfacce grafiche

Per un migliore controllo degli alert è fortemente consigliato l'utilizzo di alcune interfacce grafiche.

Le più comuni e con le quali si ottengono migliori prestazioni sono ACID, BASE (basato su ACID) , WEBMIN, Snort Grok, Sguil.

ACID

Analysis console of intrusion databases, é un motore di analisi basato su un motore di analisi in PHP, che cerca e processa gli eventi generati dai vari IDS, firewall e monitor di rete, nei database di sicurezza. Le caratteristiche attualmente includono:

- Query-builder e interfaccia di ricerca per trovare le segnalazioni di allarmi all'interno di meta informazioni come firma, tempo di rilevamento, nonché le informazioni di rete come origini/destinazione, porte e flag.
- Packet-viewer (decoder) mostra graficamente le informazioni dei pacchetti riguardanti il layer-3 e il layer-4, contenute negli alert loggati.
- Gestione degli alert, raggruppandoli e provvedendo ad eliminare falsi positivi e falsi negativi, provvede inoltre ad inviare un email ogni qualvolta venga rilevato un nuovo alert.
- Generazione di grafici e statistiche basate sul tempo, sui sensori, sulle firme, sui protocolli, sugli indirizzi IP, sulle porte TCP/UDP.

ACID ha la capacità di analizzare una vasta gamma di eventi che vengono poi post-processati all'interno del suo database.

Per quanto riguarda la sua installazione richiede che venga copiata la cartella Acid, presente all'interno del pacchetto scaricato dal sito **http://www.andrew.cmu.edu/user/rdanyliw/snort/snortacid.html**, all'interno della document-root di Apache.

In seguito modificare il file di configurazione “acid_conf.php” come segue:

```
o $DBlib_path     : directory di installazione di ADODB


o $DBtype         : tipo di database usato, in questo caso “mysql”

o $alert_dbname   : nome del database degli alert
o $alert_host     : nome del server del database degli alert
o $alert_port     : porta nella quale é memorizzato il database
o $alert_user     : username per il database degli alert
o $alert_password : password
```

BASE

Basic Analysis and Security Engine, si basa sul codice del progetto ACID.

Questa applicazione fornisce un front-end Web delle query e analizza le segnalazioni provenienti da un sistema Snort IDS.

È un’interfaccia web per eseguire analisi delle intrusioni che Snort ha rilevato sulla rete. Utilizza l’autenticazione degli utenti, in modo che l’amministratore può decidere quali e quanti informazioni ogni utente può visualizzare. È molto semplice da usare, e si può effettuare la modifica dei file in maniere diretta.

Anche per BASE l’installazione consiste nello spostare la cartella principale all’interno della directory document-root di Apache.

Per la configurazione è possibile editare il file base_conf.php.dist, rinominarlo in base_conf.php, e all’interno del quale bisogna modificare i seguenti paramentri:

$BASE_urlpath = "/base"; di default all’interno delle “” non ci sarà nulla, va quindi inserito “base”

$DBlib_path = "../adodb"; va inserita la directory di ADODB

$DBtype = "mysql"; anche qui il campo sarà vuoto è in questo caso va inserito “mysql”

$alert_dbname = "snort"; qui andrà il nome del database precedentemente creato

$alert_host = "localhost"; questo è il nome della macchina su cui si trova il database

$alert_port = "";

$alert_user = "***** USER NAME *****"; va inserito il relativo username del database e sotto la password

$alert_password = "***** USER PASSWORD *****";

Per la creazione di grafici che rappresentino in maniera più chiara e semplice il traffico "sniffato" da Snort vengono usate delle librerie grafiche apposite per BASE. Esse però necessitano a loro volta di Pear.

Pear, PHP Extension and Application Repository, è come dice il nome, un repository di software in codice PHP, che mira a fornire delle librerie di codice strutturato, un sistema per la distribuzione di codice e per la gestione dei pacchetti, e promuove inoltre un codice standard per la codifica.

Prima di procedere con l'installazione di Pear assicurarsi di avere il pacchetto PHP.

Esistono due maniere diverse di scaricare sulla propria macchina Pear.

È possibile utilizzare un browser testuale chiamato Linx, se esso non è presente sul computer sarà necessario procurarsi anche quest'ultimo, semplicemente scrivendo sulla linea di comando:

```
yum install lynx
```

Una volta effettuato il download e l'installazione, si avrà modo, utilizzando l'applicazione appena caricata di reperire anche Pear.

Basterà scrivere nella console:

```
lynx –source http://pear.php.net/go_pear | php
```

Alternativamente se non si vuole dover ricorrere a Lynx i comandi da eseguire sono:

```
wget http://pear.php.net/go_pear

mv go_ pear go_pear.php

php go_pear.php
```

Quest'ultimo commando server ad eseguire il programma.

Per caricare le librerie grafiche che permetto a BASE di produrre grafici inserire nella Shell il comando:

```
./pear install Image_Graph-alpha Image_Canvas-alpha Image_Color
Numbers_Roman
```

Ora non rimane che collegare il database all'interfaccia grafica. Per poterlo fare è necessario ancora un'applicazione, ADODB.

In questa occasione si utilizzerà come UI, BASE, quindi si procede con l'installazione di ADODB.

Il pacchetto è reperibile sul sito sourgeforge.org.

La cartella che si otterrà può essere collocata dove si vuole. La cosa importante è poi inserire la giusta directory nei file di configurazione dell'eventuale interfaccia grafica, ACID o BASE.

Webmin

La gestione da remoto di un server Linux può risultare problematica per chi ha poca dimestichezza con strumenti come SSH e la riga di comando. Con Webmin tutto diventa più semplice: creare nuovi utenti, condividere file, pianificare i backup, impostare le interfacce di rete, ma anche configurare Apache o MySQL. Il software fornisce un'intuitiva interfaccia web per le più consuete operazioni d'amministrazione di un sistema Unix. Basterà un moderno browser per gestire il server.

Webmin in sintesi è costituito da un mini web server, al quale è possibile collegarsi dalla porta 10000, e da un certo numero di script Cgi. Questi ultimi agiscono modificando i file di configurazione normalmente modificati a mano. Il software, ideato e creato da Jamie Cameron (che rimane tuttora il principale sviluppatore), è scritto in Perl quindi Perl 5 rappresenta un requisito indispensabile per la sua installazione.

Webmin viene distribuito con una licenza stile BSD che consente non solo di modificarlo liberamente, ma anche di includerlo in applicazioni commerciali. Ha una struttura modulare che ne permette l'espansione con software scritto da terze parti. I moduli standard, compresi nell'installazione di base, sono sufficienti per le più comuni esigenze.

Per l'installazione si possono percorrere due strade: prelevare il pacchetto precompilato dal sito o configurare YUM per collegarsi al repository di Webmin.

Nel primo caso si deve scaricare sulla propria macchina la versione attualmente disponibile, *webmin-1.450-1.noarch.rpm,* assumendo i privilegi di root e nel prompt si deve digitare:

```
[root]# rpm -ivh webmin-1.450-1.noarch.rpm
```

Al termine del processo, che si occuperà di segnalare eventuali dipendenze, comparirà un messaggio di questo tipo:

```
Webmin install complete. You can now login to
https://my.host.name:10000/
as root with your root password.
```

Da cui si intuisce come l'amministrazione del server avverrà collegandoci su connessione cifrata SSL all'hostname della macchina, *host1.naso.eu* nell'esempio. In alternativa potremmo usare il suo indirizzo IP o, se ci colleghiamo localmente, 127.0.0.1 o localhost.

Si noti subito che il web server integrato si metterà in ascolto sulla porta 10000**,** per non infastidire il web server standard solitamente attivo sulle porte 80 e 443. Da tenere presente questo particolare se si deve configurare eventuali firewall posti a protezione della macchina. Infine si viene informati che per gestire l'applicazione bisognerà essere autenticati con le credenziali di root.

Viene mostrato ora al metodo alternativo, basato su YUM, che prevede la creazione di un semplice file di configurazione *webmin.repo*:

```
[root]# cd /etc/yum.repos.d
[root]# touch webmin.repo
```

Una volta aperto il file appena creato con un editor di testi vanno inserite le righe seguenti:

```
[Webmin]
name=Webmin Distribution Neutral
baseurl=http://download.webmin.com/download/yum
enabled=1
```

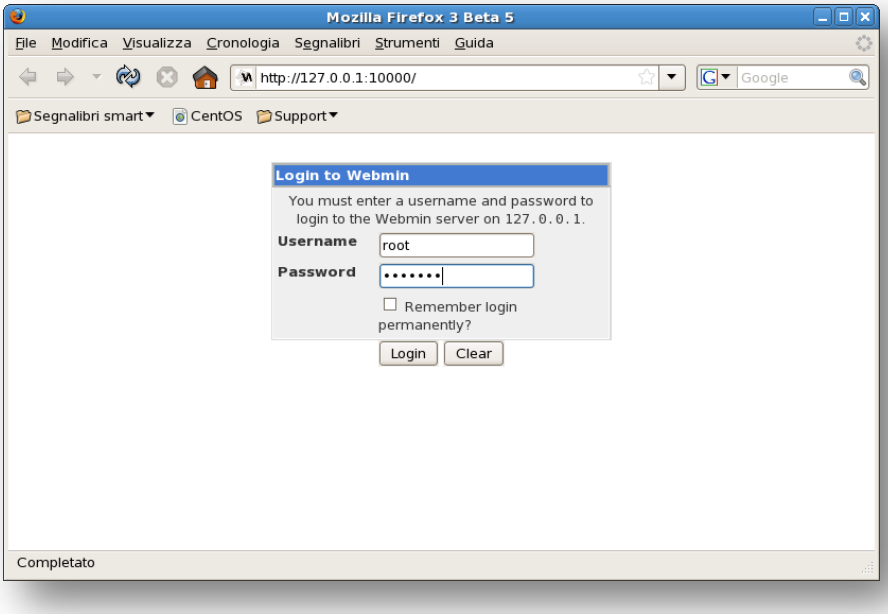
Ora si importerà la chiave GPG con cui il pacchetto sarà firmato:

```
[root]# rpm --import http://www.webmin.com/jcameron-key.asc
```

terminando con l'installazione vera e propria:

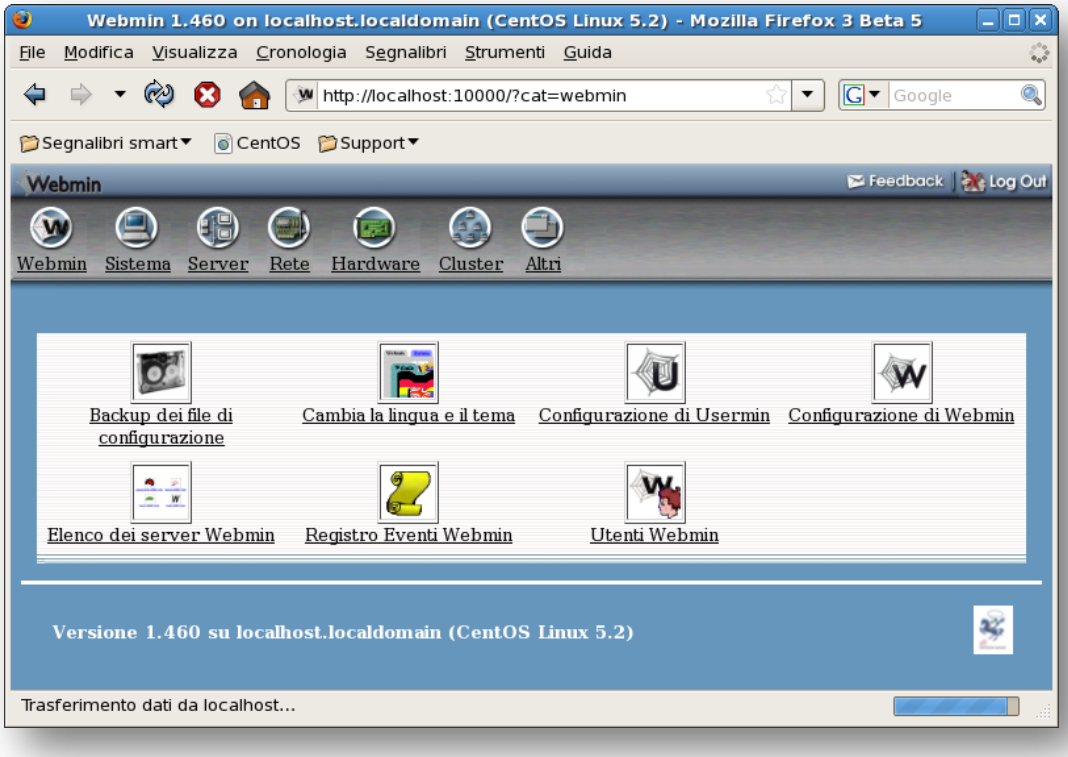
```
[root]# yum install webmin
```

Giunti a questo punto non resta che collegarsi e testare le funzionalità del programma. Aprendo il browser e digitando l'indirizzo https://host1.naso.eu:10000/ si accederà alla pagina principale del programma.



Nella schermata di login vanno inseriti nome utente e password di root per entrare. La pagina iniziale proporrà subito l'aggiornamento di alcuni moduli, ovviamente ciò dipenderà dalla versione del software installata e da eventuali successivi fix proposti dall'autore. Un click sul pulsante *Install Updates Now* risolverà tutto in automatico e un successivo click sul link *Return to Webmin configuration* riporterá l'utente alla pagina *Webzine Configuration*.

In questa schermata compaiono tutte le opzioni di configurazione del programma. È possibile selezionare attraverso l'icona *Language* quale lingua usare come predefinita, modificando da inglese a italiano. Poi da *Temi di Webmin* si può modificare l'aspetto grafico delle pagine. Altri temi sono disponibili alla pagina del sito che ne riporta l'elenco completo. Una volta trovato quello che desiderato sarà possibile installarlo con un upload dal file system locale o da un indirizzo http o ftp.



Passando ora agli aspetti più sostanziali si nota che il menu in alto, accorpa in macro categorie le aree del server su cui si può intervenire:

- Webmin: tutte le opzioni di configurazione che riguardano il programma e la sua modalità di funzionamento.
- Sistema: raccoglie gli aspetti generali di configurazione del server come la creazione di utenti e gruppi, l'impostazione di operazioni pianificate, il controllo dei dischi, il mount di partizioni e così via.
- Server: qui si trova l'elenco dei software server installati sulla macchina e gestibili con Webmin come SSH, Apache, Bind, Samba, Postfix.
- Altri: elenco di opzioni di configurazione non catalogabili nelle restanti categorie. Ad esempio qui è possibile modificare la configurazione di PHP.

- Rete: tutto ciò che si riferisce al networking come la configurazione delle interfacce di rete o delle route, le impostazioni del firewall e così via.
- Hardware: controlla gli aspetti fisici del server in particolare la gestione dei dischi con impostazione del RAID, LVM e simili. Consente anche la masterizzazione.
- Cluster: permette addirittura la gestione di cluster di server.
- Un-used Modules: elenco dei moduli disponibili che si riferiscono a software non installato sul nostro server. È possibile direttamente da qui installare tali programmi, per farlo Webmin si appoggerà a YUM.

Risulta evidente come non sia possibile descrivere nel dettaglio i vari moduli. Nel Wiki di Webmin si trova comunque un'abbondante documentazione che aiuterà a muovere i primi passi per la configurazione del server. Qui si avrà accesso a FAQ, Tutorial e informazioni su quasi tutti i moduli. Se poi si vorrà svilupparne di propri è disponibile anche la documentazione relativa alle API.

Modalità di Snort

Ora che tutto il necessario è stato installato e configurato si può passare a utilizzare Snort.

C'è da dire che per prima cosa bisogna far sí che Snort venga avviato come Demone, e cioè è sempre attivo, lavorando come servizio in background.

Per fare ciò digitare:

snort –i eth0 –c /etc/snort/snort.conf –D

Questo per quanto riguarda la scheda di rete orientata verso l'esterno.

Per la periferica eth1 si usa lo stesso comando, sostituendo ovviamente a eth0 il nome giusto, così si avrà l'analisi dei pacchetti entranti e uscenti da tutte e due le schede di rete.

A questo punto Snort è già attivo e sta analizzando il traffico della rete.

Come detto in precedenza ha tre modalità di funzionamento di base, più una quarta, chiamata InLine Mode che verrà analizzata in dettagli più avanti.

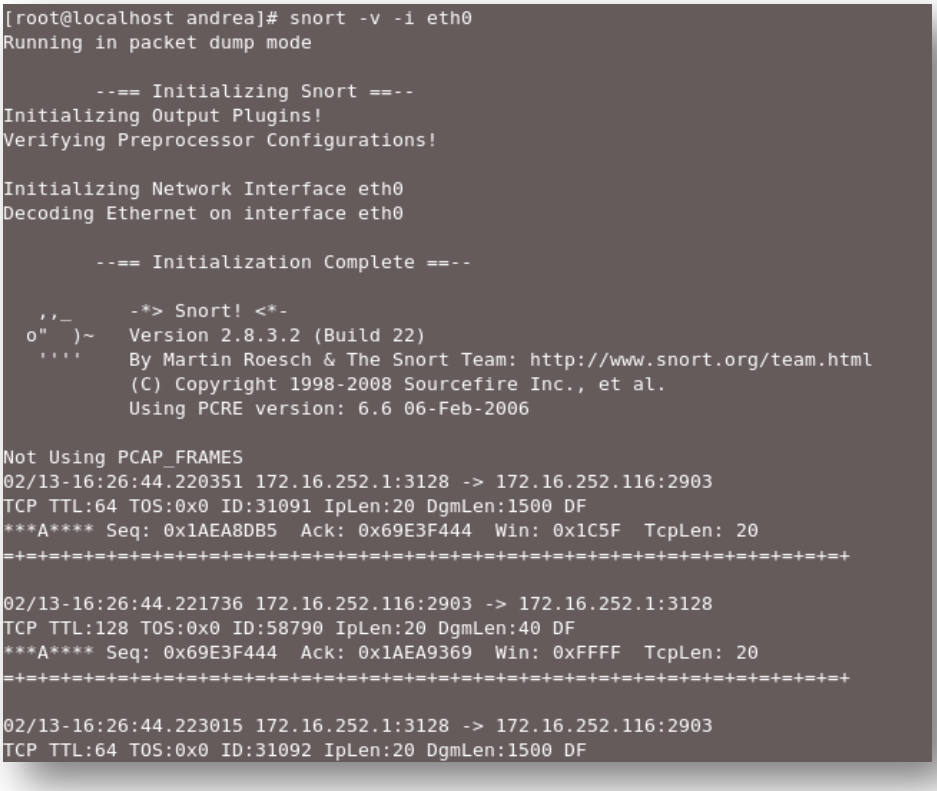
Sniffer Mode

Siccome è stato attivato su entrambe le schede di rete in Sniffer Mode non fará altro che intercettare i pacchetti, aprirli (non solo l'header) e controllare che il loro contenuto non corrisponda a un'intrusione.

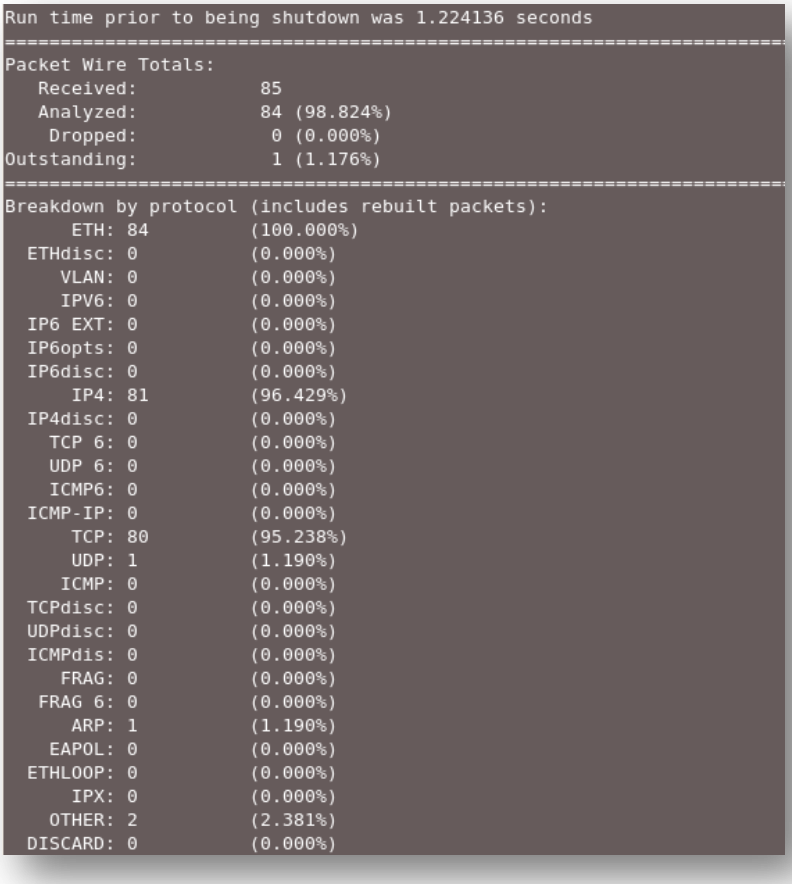
Per stampare semplicemente a video quali sono i pacchetti TCP, UDP e ICMP che transitano per la macchina:

```
./snort –v
```

Questo comando visualizzerà l'indirizzo IP fonte del pacchetto, il tipo, cioè TCP, UDP o ICMP, e l'header e nient'altro. La v del comando sta per "verbose", e ciò significa che verrà prodotto un output ingente.



Alla chiusura di Snort, il programma fornire un resoconto su quanto svolto, sui pacchetti ricevuti e analizzati:



Se si vuole avere una visione più dettagliata del traffico allora si userà:

./snort –vd

Ciò permetterà di visualizzare oltre che gli header, anche il contenuto dei dati.

Per un ulteriore precisione dell'output:

./snort –vde

mostrerà, oltre a ciò che è stato detto prima si ha la possibilità di vedere le informazioni dell'header del secondo layer.

Packet Mode

In questa modalità Snort oltre ad intercettare il traffico, fa anche il log dei contenuti di questi ultimi sul disco rigido, e tutti quelli che vengono ricevuti vengono organizzati in una directory /log, in base all'IP di provenienza.

Il comando per entrare in tale modalità è:

./snort –vde –l /var/log/snort

Se non dovesse essere presente la directory Snort uscirà, e segnalerà un errore con un messaggio, quindi è necessario crearne una prima di eseguire l'applicazione.

Per "loggare" i pacchetti soltanto di una determinata rete basterà specificare il range di indirizzi utilizzati in essa:

./snort –vde –l /var/log/snort 192.168.1.0/24

Se si dovesse essere in presenza di una rete ad alta velocità che renderebbe quasi impossibile la consultazione del traffico, si potrebbe utilizzare il comando:

./snort –vde –l /var/log/snort –b

Il –b é utilizzato per salvare in maniera compatta i dati. Questa modalità di log è chiamata Binary Mode.

Una volta che i pacchetti sono stati salvati è possibile consultarli con l'opzione –r:

./snort –dvr packet.log

Si può anche decidere cosa visualizzare:

./snort –dvr packet.log icmp

In questo modo verranno mostrati soltanto i pacchetti ICMP ad esempio.

NIDS – Network Intrusion Detection System Mode

Per avviare snort in questa modalità il comando da digitare in console sarà:

./snort –dev – l /var/log/snort –h 192.168.1.0/24 –c snort.conf

Dove snort.conf é il file all'interno del quale devono essere inserite le regole di Snort, il comando applicherà queste ultime.

Se non viene specificata una particolare directory per il logging verrà utilizzata quella di default che è /var/log/snort.

L'opzione –v (verbose mode) non dovrebbe essere utilizzata, in quanto la stampa a video, soprattutto da Bash(Shell di CentOS), è un'operazione lenta e dispendiosa in termini di cicli di CPU, e potrebbe inoltre comportare la perdita di alcuni pacchetti.

Riassumendo:

./snort –d –h 192.168.1.0/24 –l /var/log/snort –c snort.conf

Questo é il modo di avviare Snort in NIDS Mode, e avvierà il processo nella sua forma base, e cioè con:

- Log attivo
- Regole di Snort in snort.conf
- Testo chiaro in ASCII
- Log in directory

NIDS – OUTPUT Options

Ci sono diversi modi di configurare l'output di Snort in NIDS mode.

Il modo di default di log e di ALERT consiste nel loggare tutto in formato ASCII e usare Full Alerts.

Il FULL ALERT mode oltre a stampare gli headers dei pacchetti stampa anche dei messaggi di ALERT.

Esistono sette tipi di ALERT MODE:

- FULL (-A full)
- FAST (-A fast)
- SOCKET (-A unsock)
- SYSLOG
- CONSOLE (-A console)
- CMG (-A cmg)
- NONE (-A none)

FAST : scrive l'alert in un formato semplice.

Scrive tempo, messaggi di ALERT, l'indirizzo IP sorgente del pacchetto e quello di destinazione specificando anche la porta.

FULL : È l'alert mode di default.

UNSOCK : Manda gli alert a una socket UNIX in ascolto da un altro programma.

NONE : Disabilita l'Alert mode

CONSOLE : Manda gli alert alla console in formato fast

CMG : Genera gli alert in formato CMG

SYSLOG : Per mandare gli alert al syslog è necessario usare l'opzione -s

I Preprocessori

I preprocessori, sono dei plug-in di Snort che analizzano il comportamento dei pacchetti. Ogni preprocessore ha la funzione di identificare una diversa tipologia di attacco. Qualora il comportamento dei pacchetti dovesse risultare dannoso, essi vengono inviati al detection engine che

provvederà a verificarne il pattern matching con le regole. I preprocessori possono essere attivati, disattivati e configurati attraverso il file /etc/snort.conf.

Per capirne meglio il funzionamento, si esaminano i preprocessori HTTPInspect e sfportscan e le loro principali opzioni di configurazione.

Il preprocessore HTTPInspect, si occupa di decodificare il traffico HTTP e di identificare attacchi a livello applicativo che sfruttano eventuali vulnerabilità del protocollo HTTP. La configurazione di questo preprocessore è divisa in due parti, una globale e una per i server.

La configurazione globale è identificata dalla stringa:

```
preprocessor http_inspect: global [opzioni di configurazione]
```

I parametri che possono essere configurati sono:

- iis_unicode_map [filename (located in the config dir)] [codemap (integer)]

che deve essere sempre specificato in quanto contiene la global IIS unicode map.

- detect_anomalous_servers

che genera un allarme se viene rilevato traffico HTTP su porte non standard; è opportuno non attivare questa opzione se è prevista una configurazione server di default.

La sezione dedicata ai server ha due modalità di configurazione: default e IP. La stringa che identifica la configurazione di default è:

```
preprocessor http_inspect_server:
server default [server options]
```

mentre quella IP, che identifica la configurazione di indirizzi IP individuali, è:

```
preprocessor http_inspect_server:
server [IP] [server options]
```

Le opzioni specificabili sono:

- profile [all/apache/iis]

che permette di selezionare dei profili predefiniti in base al tipo di server HTTP utilizzato, scegliendo tra ‘all’, ‘apache’ e ‘iis’. Questa opzione può essere combinata ad opzioni come ‘ports’, ‘iis_unicode_map’, ‘flow_depth’, ‘no_alerts’, ‘inspect_uri_only’ e ‘oversize_dir_length’ che vanno specificate dopo il profilo in questo modo:

```
preprocessor http_inspect_server: server 1.1.1.1 profile all ports { 80 3128 }
```

- ports { [port] [port] . . . }

che indica su quale porta è attivo il servizio HTTP. Il traffico cifrato SSL non potrà essere decodificato.

- flow_depth [integer]

che specifica quanti byte del payload di risposta del server ispezionare. Questa opzione incrementa notevolmente le prestazioni dell’IDS perché permette di ignorare una buona parte del traffico HTTP. Il valore può essere impostato da -1 a 1460. -1 permette di ignorare l’intero traffico di risposta, mentre 0 permette di ispezionare integralmente i payload dei pacchetti.

- ascii [yes/no]

che permette di decodificare un URL che contiene sequenze di caratteri ASCII.

- utf_8 [yes/no]

che permette di decodificare un URL che contiene sequenze di caratteri UTF-8.

- iis_unicode [yes/no]

che permette di usare la mappa di default, se non è specificata una IIS Unicode Map.

- multi_slash [yes/no]

che permette di generare un allarme ogni volta che viene incontrata una stringa contenente più caratteri '/' consecutivi. (Es. "pippo/////////pluto")

- iis_backslash [yes/no]

che permette di generare un allarme ogni volta che viene incontrata una stringa contenente un carattere '\'. (Es. "pippo\pluto")

- no_alerts

che permette di non ricevere allarmi generati da questo preprocessore. Se questa opzione viene selezionata, le rispettive regole HTTP non hanno alcun effetto.

- oversize_dir_length [non-zero positive integer]

che specifica la lunghezza massima di un URL. Generalmente la lunghezza media è di 300 caratteri.

- inspect_uri_only

che migliora notevolmente le prestazioni in quanto permette di esaminare solo la porzione di payload contenente l'URL.

Un esempio di configurazione del preprocessore HTTPInspect è:

```
# unicode.map should be wherever your snort.conf lives, or
# given a full path to where snort can find it.
preprocessor http_inspect: global \
iis_unicode_map unicode.map 1252
preprocessor http_inspect_server: server 1.1.1.1 \
ports { 80 3128 8080 } \
flow_depth 0 \
ascii yes \
oversize_dir_length 300
```

Dal precedente codice si deduce che il file contenente la mappa Unicode è unicode.map, l'indirizzo IP del server HTTP è 1.1.1.1 il quale è attivo sulle porte 80,3128 e 8080. Non sarà ispezionato il payload dei pacchetti di risposta del server, ma saranno decodificati gli URL contenenti caratteri ASCII che potranno avere una lunghezza massima di 300 caratteri.

Il preprocessore sfportscan invece, si occupa di identificare la prima fase di un attacco, dove l'attaccante cerca di acquisire informazioni sui protocolli e sui servizi supportati da una vittima. Questo preprocessore, permette di identificare qualsiasi tipo di Portscan.

A tal proposito, è necessario che sia abilitato il preprocessore flow con il quale il preprocessore sfportscan interagisce, mediante la seguente stringa:

```
preprocessor flow: stats_interval 0 hash 2
```

I parametri che possono essere configurati per il preprocessore sfportscan sono:

- proto { <proto> }

che può essere configurato con una delle seguenti opzioni: 'tcp', 'udp', 'icmp', 'ip' oppure 'all' se si intende esaminare tutti i protocolli.

- scan_type { <scan_type> }

che può essere configurato con le opzioni: 'portscan', 'portsweep', 'decoy_portscan', 'distributed_portscan' oppure 'all' se si intende monitorare tutti i tipi di scan.

- sense_level { <level> }

che accetta i parametri: 'low', 'medium' o 'high' in base al grado di sensibilità che si vuole assegnare al sensore.

- ignore_scanners { <ip_list> }

che definisce gli indirizzi IP che possono eseguire scansioni e pertanto da ignorare.

- ignore_scanned { <ip_list> }

che definisce gli indirizzi IP che possono ricevere scansioni e pertanto da ignorare.

- logfile { <file> }

che definisce su quale file salvare l'output delle scansioni.

Un esempio di configurazione è:

```
preprocessor sfportscan: proto { all } \
scan_type { all } \
logfile { /var/log/snort/portscan } \
sense_level { high }
```

Dal precedente codice si deduce che saranno esaminati i pacchetti appartenenti a tutti i protocolli, saranno monitorati tutti i tipi di scansioni, il file contenente i log dei Portscan sarà /var/log/snort/portscan, e il sensore avrà una sensibilità alta.

Il Detection Engine

Il Detection Engine è il componente che riceve i pacchetti dai preprocessori e si occupa di confrontarli con le regole d'intrusion detection. Nel caso in cui dovesse esserci una corrispondenza tra un pacchetto e più regole diverse, la prima regola che trova una corrispondenza con il contenuto di un pacchetto genera un allarme o, in alternativa, Snort offre anche la possibilità di generare un allarme per ciascun evento. Per ridurre il numero di falsi positivi può essere configurato il file threshold.conf. Siccome ogni evento è associato ad un gen_id e un sig_id, conoscendo questi due valori, è possibile disabilitare completamente gli allarmi in questo modo:

```
# Suppress this event completely
suppress gen_id 1, sig_id 1852
# Suppress this event from this IP
suppress gen_id 1, sig_id 1852, track by_src, ip 10.1.1.54
# Suppress this event to this CIDR block
suppress gen_id 1, sig_id 1852, track by_dst, ip 10.1.1.0/24
```

Per garantire l'effettiva disattivazione delle regole, è opportuno accertarsi che il file threshold.conf sia incluso nel file snort.conf mediante la stringa

```
include threshold.conf
```

È anche possibile fare in modo che in un certo intervallo di tempo venga generato al massimo un allarme.

```
# Esempio
threshold gen_id 1, sig_id 1851, type limit, track by_src,
count 1, seconds 60
```

Comprendere i messaggi di ALERT standard

I messaggi di Alert appaiono in questa forma:

[* *] [116 : 56 : 1] (snort_decoder) : TCP/IP Detected [* *]

116 - Generator ID

Questo comunica all'utente quale componente di Snort ha generato l'Alert.

E possibile leggere la lista dei Generators IDs (GIDS) nel file etc/generators all'interno del sorgente di Snort.

116 in questo caso è il "decode" component di Snort

56 – Snort ID

Indica quale preprocessore è stato utilizzato.

Per una lista completa dei SIDS si legga: etc/gen-msg.map

Nel caso in questione 56 rappresenta un evento TCP.

1 – Revision ID

Configurare Snort per elevate performance

Se si vuole far lavorare Snort velocemente (ad esempio per linee ad un Gigabit) dobbiamo usare il logging unificato, e un lettore di log unificati come Barnyard

./snort –b –A fast –c snort.conf

Snort con GUI

Alternativamente a quanto detto finora, e anche caldamente consigliato, vi è l'utilizzo di alcune interfacce grafiche per la visualizzazione e anche la gestione del lavoro svolto da Snort.

Le più diffuse e più utilizzate sono, come già detto nel capitolo precedente, ACID, BASE e Webmin.

Verranno ora mostrati i dettagli, allo scopo di illustrare il loro funzionamento e le caratteristiche principali, mostrando poi in seguito alcune prove svolte, con l'ausilio di queste ultime.

La User Interface che è stata utilizzata per configurare al meglio Snort e monitorarlo a seconda delle esigenze richieste dalla realizzazione di questo sistema è BASE. Essendo nata da una costola di ACID è molto simile ad essa anche nella rappresentazione del tutto.

Per accedere alla finestra principale di BASE è sufficiente aprire una pagina di un qualsiasi browser e scrivere nella barra degli indirizzi :

http://localhost/base

Di seguito viene riportata la schermata d'inizio, dalla quale salta subito all'occhio la parte centrale, nella quale sono presenti delle barre vuote.

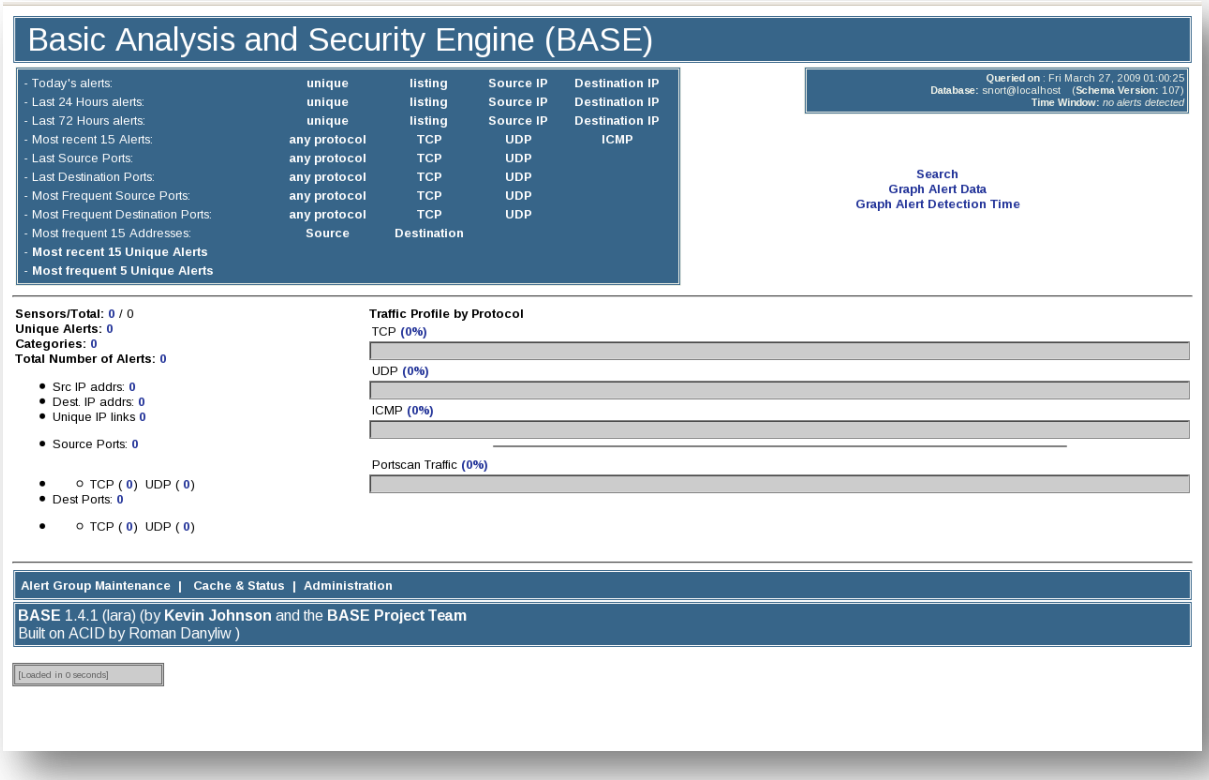
Queste barre con l'aumento del traffico di rete cominceranno a riempirsi, in base alla quantitá di pacchetti di ciascuno dei tre tipi presenti, TCP,UDP e ICMP, più la rappresentazione dei portscan effettuati alla macchina.

È possibile inoltre vedere subito di fianco alle barre la quantitá di sensori che sono attivi durante la fase di "sniffing", gli alert registrati, divisi anche in categorie.

Tramite la parte superiore della schermata sarà possibile accedere alla lista dei pacchetti analizzati, divisi e organizzati per data e per ora.

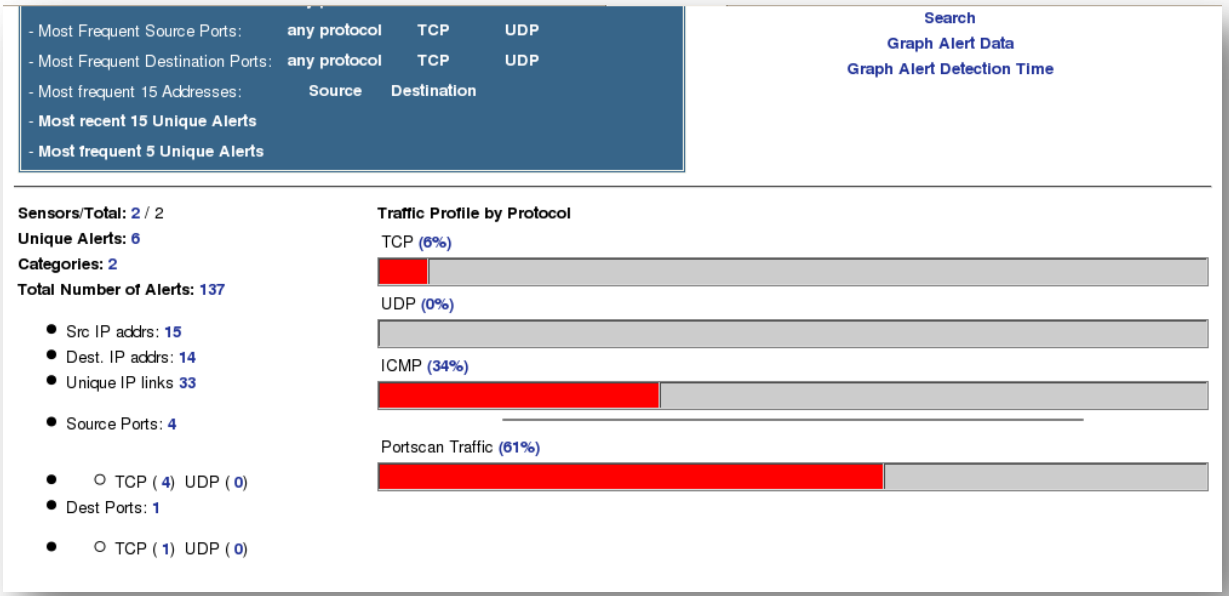
Nella parte inferiore invece sono presenti le opzioni di amministrazione di Snort e del database.

Vi è anche una possibilità di creare vari utenti con permessi e diritti diversi, in base alle esigenze.



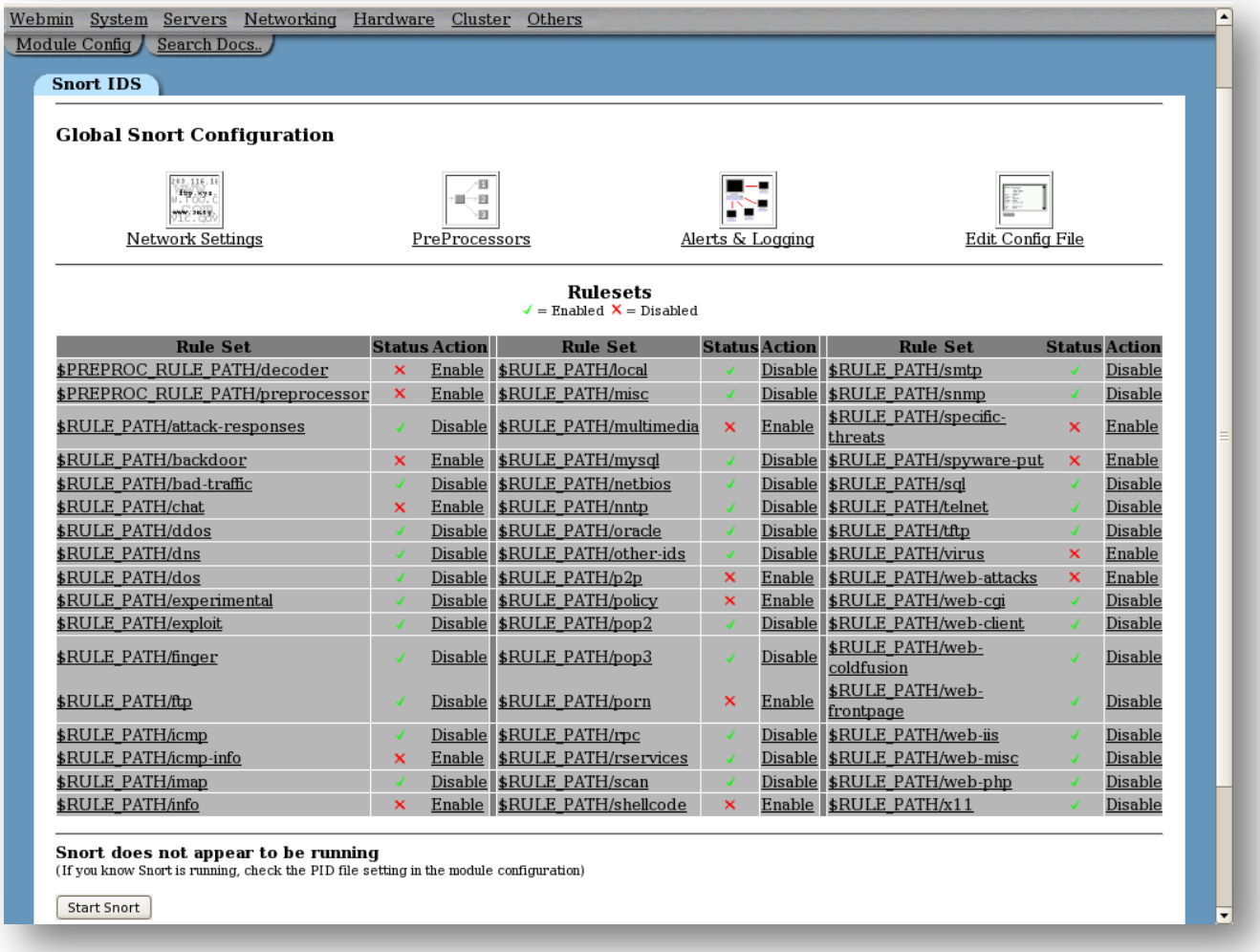
Questa schermata mostra peró le condizioni inziali, nelle quali il traffico è pari a zero. Appena inizierá il flusso di pacchetti attraverso una delle due schede di rete del sistema, si attiveranno i sensori posti su di esse, cominciando cosi ad intercettare il traffico.

Di conseguenza all'accensione dei sensori, anche le barre, che rappresentano in percentuale la quantitá di pacchetti, inizieranno a segnalare cosa sta passando attraverso la macchina.



Come si puó vedere nella schermata, in questo caso si stava effettuando , oltre a un normale, anche se minimo traffico internet, un portscan diretto alla macchina. I due sensori posti uno sulla periferica eth0 e l'altro su eth1 si sono accesi, cominciando a rilevare i pacchetti in arrivo.

In ogni caso anche senza effettuare nessun tipo di attacco, ma navigando in Internet da una qualsiasi delle macchine presenti nella sottorete, si puó testare il corretto funzionamento di BASE, aggiungendo nella cartella /etc/snort/rules, all'interno della quale sono situate le regole che Snort segue.



Nello screenshot precedente questa regola è attiva.

Il contenuto di local.rules è:

```
# $Id: local.rules,v 1.13 2005/02/10 01:11:04 bmc Exp $
# ----------------
# LOCAL RULES
# ----------------
# This file intentionally does not come with signatures.  Put your local
# additions here.
#alert tcp any any -> any any (msg:"TCP traffic"; sid:10000003;)
#alert udp any any -> any any (msg:"udp traffic"; sid:10000004;)
#alert icmp any any -> any any (msg:"icmp traffic"; sid:10000005;)
```

Questa finta regola ordina a Snort di segnalare come Alert qualsiasi pacchetto dei tipi UDP, TCP e ICMP.

Tolta Snort tornerá a rilevare soltanto i pacchetti contenenti il materiale descritto nelle altre rules

Per quanto riguarda la configurazione e la scelta delle regole che Snort deve seguire si puó utilizzare l'altra interfaccia grafica, Webmin.

L'immagine sottostante mostra come sia possibile abilitare o disabilitare tutte le regole di Snort. Si ricorda che le regole presenti in questo sistema sono quelle prelevate dal sito Internet http://www.snort.org/ , dopo aver effettuato la registrazione e quindi questa rappresenta la lista completa di quelle scritte dagli sviluppatori del programma.

Per far sí che Webmin possa dare la possibilitá di fare ció, è necessario prima accedere al sezione in alto a sinistra "Module Config", all'interno della quale va inserita la directory all'interno della quale si trovano le regole. In questo caso l'indirizzo è /etc/snort/rules.

Molto importante è ricordarsi che dopo aver apportato qualsiasi modifica al file snort.conf, sia mediante l'uso di un interfaccia come Webmin, sia a mano da riga di comando, è necessario riavviare il servizio di Snort. Se si sta utilizzando Webmin è possibile farlo direttamente dalla schermata sopra, con il pulsante in basso dello schermo. Se si sta lavorando da Bash allora il comando sará:

```
service snort restart
```

sotto vengono riportati alcuni esempi di pacchetti e attacchi rilevati:

< Signature >	< Classification >	< Total # >	Sensor #	< Source Address >	< Dest. Address >	< First >	< Last >
[local] [snort] ICMP Destination Unreachable Communication Administratively Prohibited	misc-activity	8(3%)	2	1	2	2009-03-19 18:19:28	2009-03-19 18:19:31
[local] [snort] ICMP Destination Unreachable Communication Administratively Prohibited	misc-activity	42(18%)	2	11	2	2009-03-19 17:40:53	2009-03-19 18:21:01
[snort] (portscan) TCP Portscan	*unclassified*	12(5%)	2	3	2	2009-03-19 17:58:44	2009-03-19 18:18:46
[snort] (portscan) TCP Portscan	*unclassified*	8(3%)	2	2	2	2009-03-19 18:19:47	2009-03-19 18:21:42
[snort] (portscan) Open Port	*unclassified*	67(28%)	2	4	11	2009-03-19 17:58:44	2009-03-19 18:18:46
[snort] (portscan) Open Port	*unclassified*	13(5%)	2	2	2	2009-03-19 18:19:47	2009-03-19 18:21:42
[snort] (portscan) TCP Portsweep	*unclassified*	4(2%)	1	1	2	2009-03-19 17:58:46	2009-03-19 18:07:01
[local] [snort] ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited	misc-activity	8(3%)	2	1	2	2009-03-19 18:11:55	2009-03-19 18:11:58
[snort] (http_inspect) WEBROOT DIRECTORY TRAVERSAL	*unclassified*	16(7%)	1	1	1	2009-03-19 18:17:21	2009-03-19 18:21:14
[snort] (http_inspect) WEBROOT DIRECTORY TRAVERSAL	*unclassified*	52(22%)	1	1	1	2009-03-19 18:19:13	2009-03-19 18:21:53
[snort] (http_inspect) DOUBLE DECODING ATTACK	*unclassified*	8(3%)	1	1	1	2009-03-19 18:20:40	2009-03-19 18:21:52
[arachNIDS] [local] [snort] ICMP PING NMAP	attempted-recon	2(1%)	1	1	1	2009-03-19 18:21:42	2009-03-19 18:21:42

In questo caso Snort ha rilevato pacchetti ICMP, dei PortScan effettuati con protocollo TCP, e anche un ping effettuato da NMAP.

In quello successivo sono visualizzati altre tipologie di attacco.

☐	[snort] (portscan) Open Port	*unclassified*	13(5%)	2	2	2	2009-03-19 18:19:47	2009-03-19 18:21:42
☐	[snort] (portscan) TCP Portsweep	*unclassified*	4(2%)	1	1	2	2009-03-19 17:58:46	2009-03-19 18:07:01
☐	[local] [snort] ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited	misc-activity	8(3%)	2	1	2	2009-03-19 18:11:55	2009-03-19 18:11:58
☐	[snort] (http_inspect) WEBROOT DIRECTORY TRAVERSAL	*unclassified*	16(7%)	1	1	1	2009-03-19 18:17:21	2009-03-19 18:21:14
☐	[snort] (http_inspect) WEBROOT DIRECTORY TRAVERSAL	*unclassified*	52(22%)	1	1	1	2009-03-19 18:19:13	2009-03-19 18:21:53
☐	[snort] (http_inspect) DOUBLE DECODING ATTACK	*unclassified*	8(3%)	1	1	1	2009-03-19 18:20:40	2009-03-19 18:21:52
☐	[arachNIDS] [local] [snort] ICMP PING NMAP	attempted-recon	2(1%)	1	1	1	2009-03-19 18:21:42	2009-03-19 18:21:42

Snort inline

Oltre alle tre modalitá principali viste finora, Snort ne possiede una quarta, l'InLine Mode.

Prima è stato specificato che Snort da solo, avviato nelle sue modalitá principali, sniffer, packet filter e NIDS, non blocca assolutamente il traffico, ma si limita a leggere il contenuto dei singoli pacchetti. Piú avanti si vedrá come far collaborare Snort con un programma, Guardian che al contrario bloccherà il traffico ritenuto "ostile". Adesso invece verrá studiato in particolare l'InLine Mode di Snort.

Snort avviato normalmente usa delle librerie pcap per "catturare" i pacchetti che deve analizzare.

Snort-inline non usa tali librerie ma prende il traffico direttamente da IPTables. Per fare ció è necessario caricare un modulo ip_queue e definire delle regole sullo stesso IPTables, che ridirigano il traffico sulla coda Queue.

Snort andrá a leggere il contenuto dei pacchetti direttamente da questa coda, seguendo le proprie regole e decidendo se lasciar passare o meno i pacchetti.

In questo modo svolgerá la funzione di IPS, bloccando gli eventuali attacchi.

Affinché Snort funzioni correttamente in questa modalitá occorre scaricare il codice di iptables-devel e compilarlo. All'interno di questo pacchetto si trovano le librerie libipq che consentono a snort di funzionare in modalitá in-line.

Il comando è molto semplice dato che si trovano sui repository di Yum:

yum install iptables-devel

Oltre a queste librerie sono necessarie anche le libnet, disponibili sul sito www.packetfactory.com.

Anche nel caso di InLine ci sono due metodi per conseguire l'installazione:

- Ricompilare il pacchetto originale di Snort, inserendo durante la compilazione il comando per attivare la modalitá in-line, attraverso la digitazione di:

  ./configure --enable-inline

  make

  make install

- Scaricare da Internet il paccheto con l'attivazione di Snort InLine giá effettuata, dal sito www.guardalosuinternetseilproxydeldavacknonfaicazettisennosmadonno.com

Avviare Snort INLine

Prima cosa assicurarsi che l' IP_queue module sia caricato. Per fare ció il comando è:

lsmod | grep ip_queue

se il modulo non fosse caricato allora sarebbe necessario:

modprobe ip_queue

Ora bisogna indirizzare il traffico a Snort usando QUEUE Target.

Ad esempio:

iptables –A OUTPUT –p tcp –dport 80 –j QUEUE

questo manda tutto il traffico TCP dalla porta 80 al QUEUE target, che significa ridirigere il traffico dall KernelSpace(Netfilter) allo UserSpace(Snort INLine).

Ora finalmente è possibile avviare Snort:

snort_inline –QDc /etc/snort_line/snort_line.conf –l /var/log/snort

Si possono usare le seguenti opzioni:

- -Q per prendere i pacchetti da IPTables
- -D per avviare Snort INLine in daemon mode (il PID viene creato in /var/run/snort_inline.pid)
- -c legge la seguente configurazione (file.conf)
- -l logga i pacchetti alla directory specificata.

In questa modalitá sono presenti tre tipi di regole:

- DROP fa scartare (a IPTables) e loggare i pacchetti
- REJECT fa scartare e loggare i pacchetti e manda un TCP reset(se il protocollo è TCP) o un ICMP PORT unreachable (se il protocollo è UDP)
- SDROP scarta i pacchetti senza loggare nulla.

REJECT ha due opzioni che possono essere usate per mandare TCP resets:

- Si puó usare RAW Socket (default)

  In questo caso bisogna avere un'interfaccia che ha assegnato un indirizzo IP.

  Altrimenti il pacchetto viene scartato e loggato peró il TCP RESET non puó essere spedito.

- Si puó anche usare un device fisico.

  Prende INDEV name dalla IP_queue e la usa come interfaccia su cui mandare i RESET.

Capitolo 3

Prove Effettuate

Problemi di sicurezza del sistema

Il sistema realizzato non vanta una configurazione ottimale affinché gli attacchi ricevuti vengano resi vani.

La condizione che non permette ció sta nel fatto che Snort, che dovrebbe rilevare e prevenire gli attacchi, si trova su una macchina potenzialmente esposta ad attacchi diretti. Come detto giá all'inizio sarebbe stato meglio che il computer sul quale è installato, fosse rimasto nascosto del tutto, in modo tale da essere invisibile ai malintenzionati e quindi poter svolgere meglio e indisturbato il proprio compito.

Gli attacchi che sono stati effettuati non erano peró particolarmente pesanti e quindi sono rilevati facilmente da Snort.

Attacchi effettuati

Portscan

La macchina dalla quale sono stati eseguiti gli attacchi ha un sistema operativo Windows XP con l'aggiunta del Service Pack 3. Su tale sistema è stata poi caricata , mediante VmWare Workstation, un macchina virtuale sulla quale è stato installato un altro sistema, questa volta Unix based, la distro Backtrack 3.0. Da etrambi i sistemi operativi sono stati tentati attacchi

Dal sistema Win32 sono parti per lo piú per verificare le vulnerabilitá, eseguiti con un tool specifico, chiamato Nessus.

Nessus è un programma open source rilasciato in licenza GPL di tipo client-server che tramite lo scan e l'abilitazione di plugin appositamente configurabili a seconda della tipologia di host e

vulnerabilità che si andrà ad analizzare, rileva le vulnerabilità presenti suggerendo le possibili soluzioni creando report di facile analisi in vari formati (HTML , pdf, etc etc).

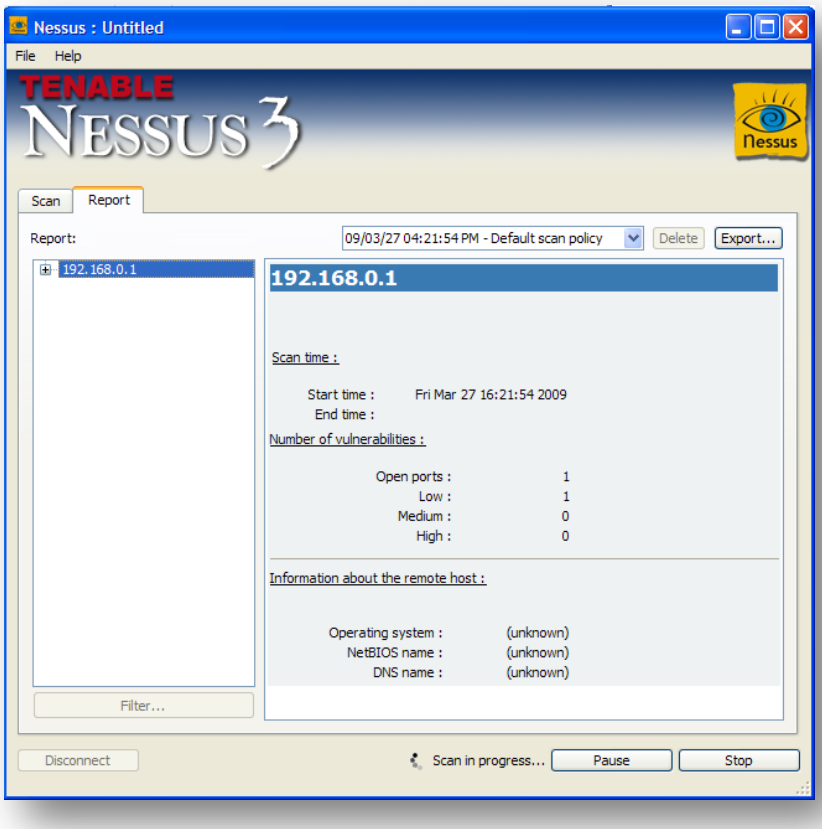
Con le sue tante opzioni per la scansione, la possibilità di scrivere plugin e per il tipo di reportistica prodotta rimane uno dei migliori strumenti per vulnerability assessment.

L'utilizzo di tale software si è rilevato molto semplice. I passi per effettuare uno scan sono pochi e semplici:

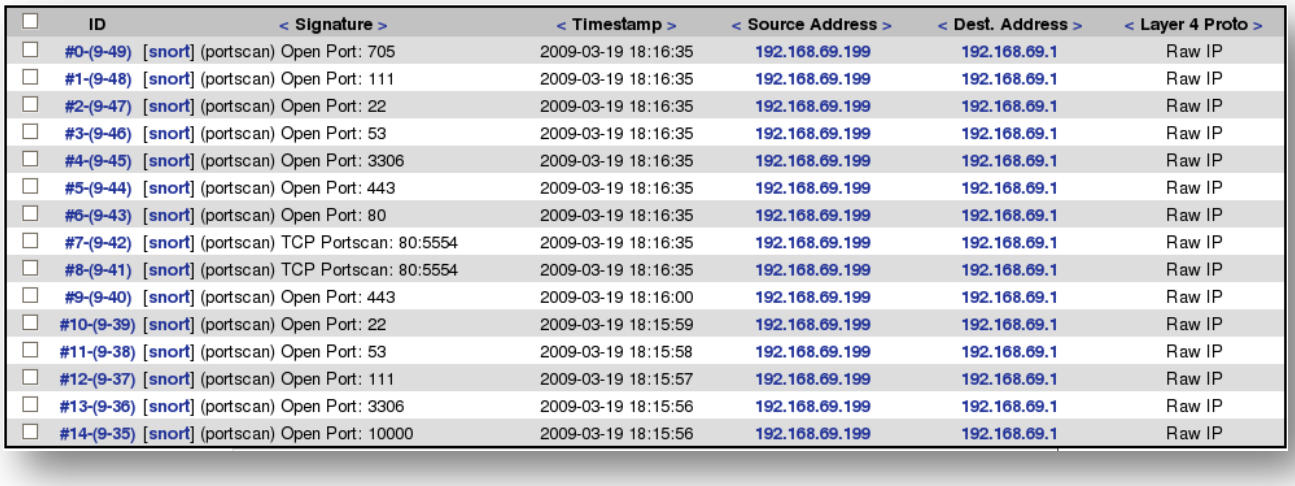
- Si crea un server in locale al quale sará necessario loggarsi prima di effettuare qualsiasi operazione
- Si inserisce uno o piú indirizzi nella casella per specificare quali host mettere alla prova.
- Una volta fatto ció ci sará da scegliere con quale politica effettuare gli scan.

  Queste politiche sono configurabili e pienamente personalizzabili. Nel caso specifico peró si è scelto di procedere con quelle messe a disposizione di default dal programma.

- Non resta che cliccare sul pulsante scan per iniziare il procedimento, dopo acluni secondi Nessus inizierá a produrre i risultati del test.



Nel frattempo sulla macchina Host1 Snort sta svolgendo il suo lavoro e ha rilevato i finti attacchi che gli sono stati sferrati:



ID	< Signature >	< Timestamp >	< Source Address >	< Dest. Address >	< Layer 4 Proto >
#0-(9-49)	[snort] (portscan) Open Port: 705	2009-03-19 18:16:35	192.168.69.199	192.168.69.1	Raw IP
#1-(9-48)	[snort] (portscan) Open Port: 111	2009-03-19 18:16:35	192.168.69.199	192.168.69.1	Raw IP
#2-(9-47)	[snort] (portscan) Open Port: 22	2009-03-19 18:16:35	192.168.69.199	192.168.69.1	Raw IP
#3-(9-46)	[snort] (portscan) Open Port: 53	2009-03-19 18:16:35	192.168.69.199	192.168.69.1	Raw IP
#4-(9-45)	[snort] (portscan) Open Port: 3306	2009-03-19 18:16:35	192.168.69.199	192.168.69.1	Raw IP
#5-(9-44)	[snort] (portscan) Open Port: 443	2009-03-19 18:16:35	192.168.69.199	192.168.69.1	Raw IP
#6-(9-43)	[snort] (portscan) Open Port: 80	2009-03-19 18:16:35	192.168.69.199	192.168.69.1	Raw IP
#7-(9-42)	[snort] (portscan) TCP Portscan: 80:5554	2009-03-19 18:16:35	192.168.69.199	192.168.69.1	Raw IP
#8-(9-41)	[snort] (portscan) TCP Portscan: 80:5554	2009-03-19 18:16:35	192.168.69.199	192.168.69.1	Raw IP
#9-(9-40)	[snort] (portscan) Open Port: 443	2009-03-19 18:16:00	192.168.69.199	192.168.69.1	Raw IP
#10-(9-39)	[snort] (portscan) Open Port: 22	2009-03-19 18:15:59	192.168.69.199	192.168.69.1	Raw IP
#11-(9-38)	[snort] (portscan) Open Port: 53	2009-03-19 18:15:58	192.168.69.199	192.168.69.1	Raw IP
#12-(9-37)	[snort] (portscan) Open Port: 111	2009-03-19 18:15:57	192.168.69.199	192.168.69.1	Raw IP
#13-(9-36)	[snort] (portscan) Open Port: 3306	2009-03-19 18:15:56	192.168.69.199	192.168.69.1	Raw IP
#14-(9-35)	[snort] (portscan) Open Port: 10000	2009-03-19 18:15:56	192.168.69.199	192.168.69.1	Raw IP

ARP Spoofing

Il secondo preso in considerazione è un attacco un pó piú complesso e prevede un tentativo di ARP Spoofing ed é stato effettuato con l'applicazione Ettercap.

Ettercap ha due principali modi di funzionare:

- In modalitá "Passivo"

  In questo modo non c'è modo di rilevare il suo operato, in quanto si rende invisibile limitandosi a "sniffare" i pacchetti in transito, e ad analizzarne il contenuto.

- In modalità "Attivo" invece lo sniffing diventa piú invasivo.

  Infatti entra in gioco la pratica chiamata ARP Poisoning.

  Essa consiste nel fatto che Ettercap fa in modo di sostituire, agli "occhi" di tutti gli host sulla sottorete, il mac address del corrispettivo Gateway, con uno fittizio definito da Ettercap stesso.

Cosi facendo succede che tutti i pacchetti destinati a raggiungere il gateway, che avrebbe provveduto a effettuare il Forwarding, finiscono per passare anche dalla macchina su cui è avviato

Ettercap, avendo cosí la possibilitá di analizzare il contenuto dei pacchetti stessi e anche di modificarli.

Anche questo attacco è stato prontamente rilevato da Snort:

< Signature >	< Classification >	< Total # >	Sensor #	< Source Address >	< Dest. Address >	< First >	< Last >
[local] [snort] ICMP Destination Unreachable Communication with Destination Host is Administratively Prohibited	misc-activity	2232(69%)	3	1	5	2009-02-04 16:09:00	2009-02-26 22:47:18
[local] [snort] TCP traffic	*unclassified*	676(21%)	1	2	2	2009-02-26 21:45:41	2009-02-26 21:46:02
[local] [snort] udp traffic	*unclassified*	37(1%)	1	11	3	2009-02-26 21:45:47	2009-02-26 21:46:01
[snort] (spp_arpspoof) Ethernet/ARP Mismatch request for Source	*unclassified*	26(1%)	1	0	0	2009-02-26 21:58:45	2009-02-26 23:18:39
[snort] (portscan) TCP Portscan	*unclassified*	21(1%)	1	2	1	2009-02-26 22:08:04	2009-02-26 22:52:40
[snort] (spp_arpspoof) Attempted ARP cache overwrite attack	*unclassified*	219(7%)	1	0	0	2009-02-26 22:08:59	2009-02-26 22:44:30
[snort] (snort_decoder): Tcp Window Scale Option found with length > 14	*unclassified*	8(0%)	1	1	1	2009-02-26 22:52:53	2009-02-26 22:52:56

Come si puó vedere,alla quarta e sesta riga nella schermata sopra, vengono segnati entrambi i tentavi di effettuare l'attacco che avrebbe previsto la modifica dell'ARP Cache, e cioè dove si trova l'indirizzo mac del gateway**.**

Nonostante questi due attacchi non siano andati a buon fine grazie a Snort, non è possibile sostenere che il sistema realizzato sia perfettamente sicuro.

Con attacchi piú complessi di quelli effettuati come test di prova, le contromisure non sarebbero bastate.

Detto ció peró, si puó asserire che comunque, il sistema creato vanta un discreto livello di sicurezza.

Capitolo 4

Altri Tools

FwSnort

FwSnort è un piccolo software che permette di tradurre le regole di Snort in regole per il firewall Iptables.
Questo consente di bloccare tutto il traffico che corrisponde a tali regole e che, di conseguenza, non è considerato sicuro.
Il punto debole di tale programma sta nella sua esecuzione.
Esso non lavora in modalità demone e questo non permette di aggiornare le regole del firewall in tempo reale, nel momento in cui un attacco si verifica.
Il funzionamento di FwSnort è il seguente:
Si avvia il programma.
Esso traduce le regole di Snort e crea uno script .sh che, se eseguito, provvederà ad aggiungere le regole al firewall.
Per ripristinare lo stato precedente del firewall è sufficiente utilizzare il seguente comando:

```
# service iptables restart
```

Oppure

```
# /etc/init.d/iptables restart
```

In questo modo si provvede a riavviare il servizio iptables e a ricaricare le sole regole scritte in /etc/sysconfig/iptables. Le regole precedentemente aggiunte da FwSnort non vengono ricaricate, in quanto lo script .sh non modifica i file di iptables.

Esempio:

```
[root@localhost Desktop]# fwsnort
=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=-=
    Snort Rules File          Success   Fail      Ipt_apply Total

[+] backdoor.rules            65        11        65        76
[+] ddos.rules                18        14        18        32
[+] dns.rules                 19        2         19        21
[+] dos.rules                 9         7         9         16
[+] exploit.rules             36        46        36        82
                              =======================================
                              147       80        147       227

[+] Generated iptables rules for 147 out of 227 signatures: 64.76%
[+] Found 147 applicable snort rules to your current iptables
    policy.

[+] Logfile: /var/log/fwsnort.log
[+] iptables script: /etc/fwsnort/fwsnort.sh
[root@localhost Desktop]#
```

Normalmente, fwsnort prende le regole da /etc/fwsnort/snort-rules.

Per una migliore visualizzazione dell’esempio, è stata tolta la maggior parte delle regole presenti in tale percorso.

Come è possibile notare, il programma ha generato lo script /etc/fwsnort/fwsnort.sh.

Eseguendo tale script verranno aggiunte 147 regole alle tabelle di iptables.

Qui di seguito viene mostrato un esempio. Per rendere ulteriormente semplificata la spiegazione, è stato utilizzato un solo file di regole: backdoor.rules.

```
[root@localhost ~]# fwsnort
=====================================================================
Snort Rules File          Success   Fail      Ipt_apply Total

[+] backdoor.rules            65        11        65        76
=======================================
65        11        65        76

[+] Generated iptables rules for 65 out of 76 signatures: 85.53%
[+] Found 65 applicable snort rules to your current iptables
policy.

[+] Logfile: /var/log/fwsnort.log
[+] iptables script: /etc/fwsnort/fwsnort.sh

[root@localhost ~]# /etc/fwsnort/fwsnort.sh
[+] Adding backdoor rules.
Bad argument `version|3A| GOLD 2.1'
Try `iptables -h' or 'iptables --help' for more information.
Bad argument `version|3A| GOLD 2.1'
Try `iptables -h' or 'iptables --help' for more information.
Rules added: 130
[+] Finished.
```

Sono state generate 65 regole applicabili al firewall.

Successivamente viene eseguito lo script, e le regole vengono effettivamente applicate.

Attraverso l'utilizzo del seguente comando:

```
# iptables –L
```

é possibile verificare le regole applicate.

Guardian

Come precedentemente detto, fwsnort ha un difetto. Non può essere avviato in modalità demone.
A risolvere questo problema interviene Guardian, un software analogo che però ha la particolarità di aggiornare in tempo reale le regole del firewall.

Installazione
L'installazione di Guardian è leggermente macchinosa.
Una volta scaricato ed estratto l'archivio compresso, è necessario seguire i seguenti passi:

- Apportare le opportune modifiche al file guardian.conf
- Creare il file /etc/guardian.ignore. Questo file serve a contenere quegli indirizzi fidati e di cui ignorare il traffico. Buona norma è inserirvi gli indirizzi dei DNS e Gateway.
- Copiare lo script perl guardian.pl in un percorso per l'esecuzione. Ad esempio /usr/bin o /usr/local/bin.
- Copiare guardian_block.sh e guardian_unblock.sh script nello stesso percorso.

A questo punto è già possibile usare guardian.
Non occorre specificare che questo programma necessita della presenza di un IDS e, attualmente, supporta unicamente Snort.

Per avviare il demone:

```
# guardian.pl –c /etc/guardian.conf
```

A questo punto, al rilevamento dell'attacco, guardian legge l'alert generato da Snort e lo traduce in una regola per il firewall. In questo modo, un successivo attacco analogo verrà immediatamente bloccato.

Anche in questo caso, per ripristinare lo stato del firewall è necessario eseguire il comando:

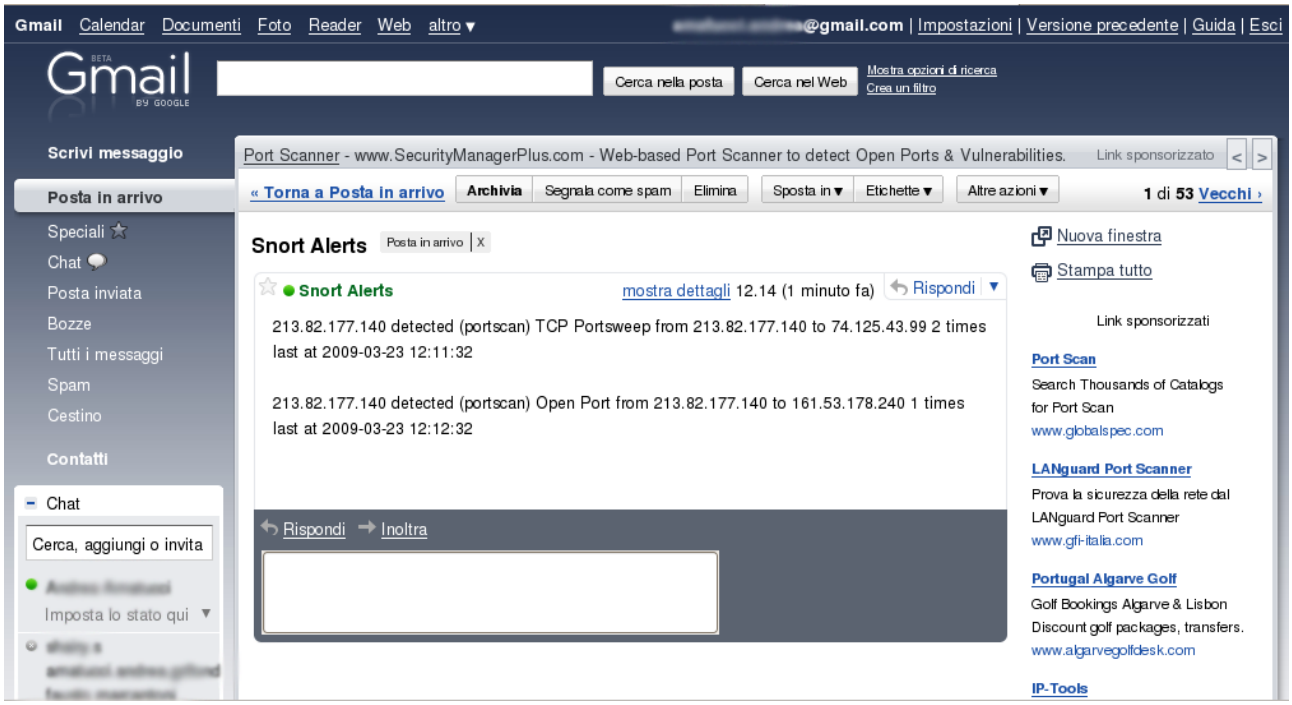
```
# service iptables restart
```

Snortnotify

Snortnotify è un tool utilizzato per inviare email ad un amministratore di rete in seguito ad un alert avvenuto nel sistema.

Questo programma funziona sfruttando cron, un programma delle piattaforme Unix che viene utilizzato per attivare comandi o script automaticamente in una data o ora specifiche, esso effettua una ricerca all'interno del database di Snort alla ricerca di nuovi alerts.se ne trova provvede ad inviare una mail con all'interno il nome del sensore che ha rilevato l'attacco, la firma e le informazioni sull'orario a cui è avvenuta la tentata intrusione.

Lo Screenshot sotto prende in esempio una email inviata da Snortnotify inseguito ad un attacco ricevuto:



Snortgrok

Snortgrok è un'interfaccia per Snort per l'interazione con PHP, MySQL.

Con questo programma si puó avere una piú chiara idea dei report di rete e delle intrusioni.

I dati vengono ordinati per data, indirizzo IP, porta, ecc…

È molto semplice da installare e per la configurazione è sufficiente impostare la password di MySQL, con la quale poi accederá al database per la riorganizzazione dei dati.

Molto importante è il fatto che non abbia bisogno di alcuna libreria grafica per funzionare, cosa che riduce di molto i tempi di installazione e configurazione.

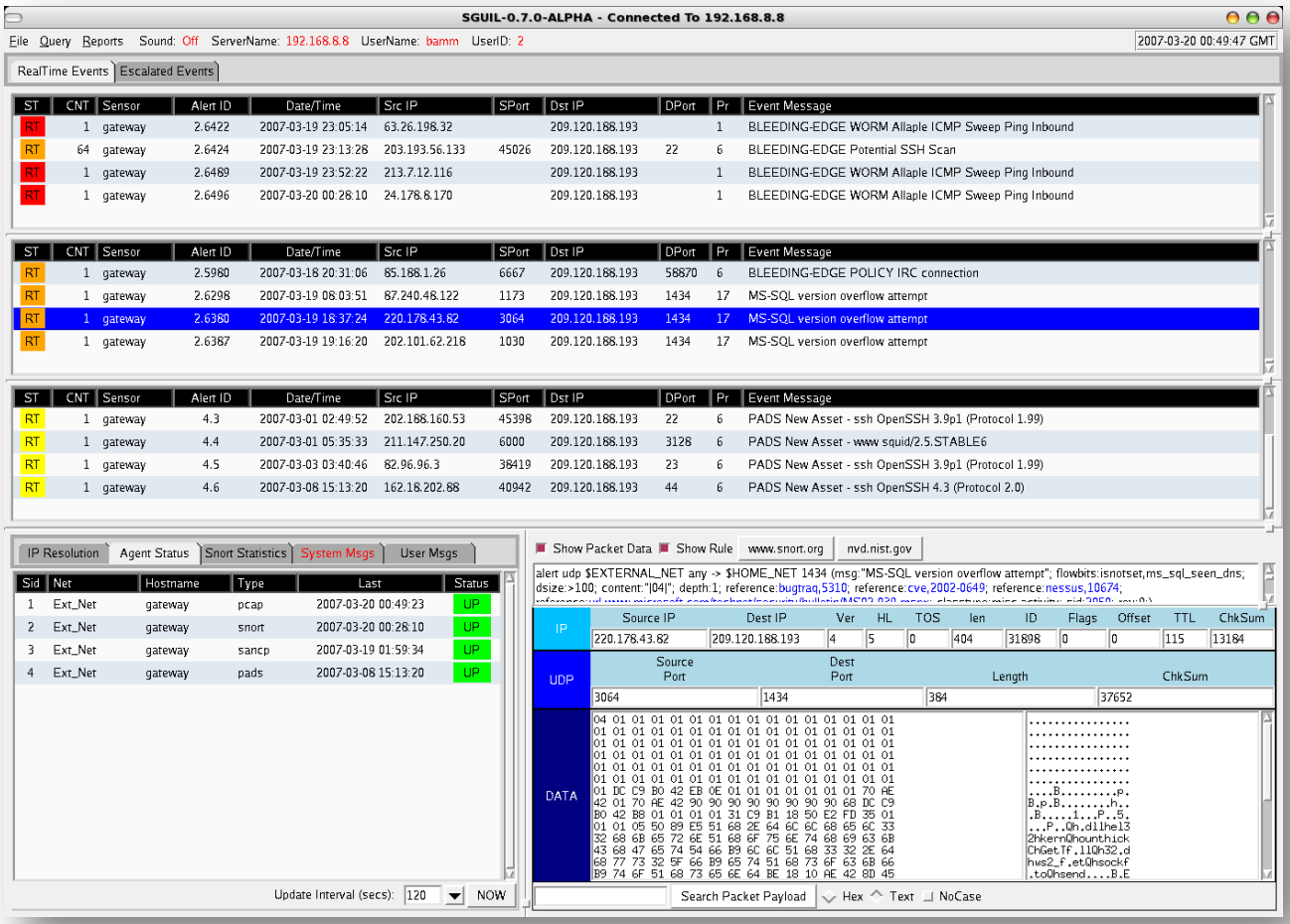
Offre inoltre un utilissimo sistema per eliminare facilmente i falsi positivi.

Aiuta anche l'utente a configurare le regole di Snort e bloccare o limitare le regole del traffico

Sguil

Sguil è un pacchetto per il monitoraggio e l'analisi della rete e degli alerts degli IDS. È stato svilupaato da analisti della sicurezza di rete per analisti della sicurezza di rete. La componente principale è un'intuitiva interfaccia grafica che permette di accedere in tempo reale agli eventi registrati in tempo reale, ai dati di sessione, e alla cattura dei pacchetti. Sguil facilita la pratica del monitoraggio della sicurezza della rete e l'analisi degli avvenimenti. Il client Sguil è scritto in Tcl/Tk e puó essere eseguito su qualsiasi sistema operativo che lo supporta, compresi Linux, BSD, Solaris, MacOS e Win32.

Sotto viene riportata una delle schermate che è possibile visualizzare durate l'uso di questo tool:



Conclusione

Dalle varie prove effettuate è venuto fuori che la realizzazione di un sistema che abbia un discreto livello di sicurezza è possibile, utilizzando esclusivamente software open source.

La maggior parte degli attacchi generici viene riconosciuto e bloccato. Le cose cambiano se si ha a che fare con attacchi mirati, che mettono a dura prova l'Intrusion Detection System.

Da questo si puó dedurre che la sicurezza informatica è pressoché irraggiungibile, nonostante i mezzi utilizzati siano tra quelli piú potenti in circolazione sulla Rete.

Un grosso lavoro per mantenere la sicurezza di una rete va svolto dall'amministratore che, oltre a tenere continui backup dei dati, deve provvedere ad effettuare repentini aggiornamenti per poter fronteggiare le minacce esterne. Fondamentale in questo senso è il continuo monitoraggio dei file di log, alla ricerca di intrusioni eventualmente non bloccate.

Appendice

Viene inserito di seguito i file di configurazione piú importante, quello di Snort.

Snort.conf

```
#--------------------------------------------------
#   http://www.snort.org     Snort 2.8.3.2 Ruleset
#     Contact: snort-sigs@lists.sourceforge.net
#--------------------------------------------------
# $Id$
#
##################################################
# This file contains a sample snort configuration.
# You can take the following steps to create your own custom configuration:
#
#  1) Set the variables for your network
#  2) Configure dynamic loaded libraries
#  3) Configure preprocessors
#  4) Configure output plugins
#  5) Add any runtime config directives
#  6) Customize your rule set
#
##################################################
# Step #1: Set the network variables:
#
# You must change the following variables to reflect your local network. The
# variable is currently setup for an RFC 1918 address space.
#
# You can specify it explicitly as:
#
# var HOME_NET 10.1.1.0/24
#
# or use global variable $<interfacename>_ADDRESS which will be always
# initialized to IP address and netmask of the network interface which you run
# snort at.  Under Windows, this must be specified as
# $(<interfacename>_ADDRESS), such as:
# $(\Device\Packet_{12345678-90AB-CDEF-1234567890AB}_ADDRESS)
#
# var HOME_NET $eth0_ADDRESS
#
# You can specify lists of IP addresses for HOME_NET
# by separating the IPs with commas like this:
#
# var HOME_NET [10.1.1.0/24,192.168.1.0/24]
#
# MAKE SURE YOU DON'T PLACE ANY SPACES IN YOUR LIST!
#
# or you can specify the variable to be any IP address
# like this:
```

```
#var HOME_NET 172.16.252.0/24
var HOME_NET $eth0_ADDRESS
# Set up the external network addresses as well.  A good start may be "any"
var EXTERNAL_NET !$HOME_NET

# Configure your server lists.  This allows snort to only look for attacks to
# systems that have a service up.  Why look for HTTP attacks if you are not
# running a web server?  This allows quick filtering based on IP addresses
# These configurations MUST follow the same configuration scheme as defined
# above for $HOME_NET.

# List of DNS servers on your network
var DNS_SERVERS $HOME_NET

# List of SMTP servers on your network
var SMTP_SERVERS $HOME_NET

# List of web servers on your network
var HTTP_SERVERS $HOME_NET

# List of sql servers on your network
var SQL_SERVERS $HOME_NET

# List of telnet servers on your network
var TELNET_SERVERS $HOME_NET

# List of snmp servers on your network
var SNMP_SERVERS $HOME_NET

# Configure your service ports.  This allows snort to look for attacks destined
# to a specific application only on the ports that application runs on.  For
# example, if you run a web server on port 8081, set your HTTP_PORTS variable
# like this:
#
# portvar HTTP_PORTS 8081
#
# Ports you run web servers on
portvar HTTP_PORTS 80

# NOTE:  If you wish to define multiple HTTP ports, use the portvar
# syntax to represent lists of ports and port ranges.  Examples:
## portvar HTTP_PORTS [80,8080]
## portvar HTTP_PORTS [80,8000:8080]
# And only include the rule that uses $HTTP_PORTS once.
#
# The pre-2.8.0 approach of redefining the variable to a different port and
# including the rules file twice is obsolete.  See README.variables for more
# details.

# Ports you want to look for SHELLCODE on.
portvar SHELLCODE_PORTS !80

# Ports you might see oracle attacks on
portvar ORACLE_PORTS 1521

# other variables
#
# AIM servers.  AOL has a habit of adding new AIM servers, so instead of
# modifying the signatures when they do, we add them to this list of servers.
```

```
var AIM_SERVERS
[64.12.24.0/23,64.12.28.0/23,64.12.161.0/24,64.12.163.0/24,64.12.200.0/24,205.18
8.3.0/24,205.188.5.0/24,205.188.7.0/24,205.188.9.0/24,205.188.153.0/24,205.188.1
79.0/24,205.188.248.0/24]

# Path to your rules files (this can be a relative path)
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\rules
var RULE_PATH /etc/snort/rules
var PREPROC_RULE_PATH ../preproc_rules

# Configure the snort decoder
# ============================
#
# Snort's decoder will alert on lots of things such as header
# truncation or options of unusual length or infrequently used tcp options
#
#
# Stop generic decode events:
#
# config disable_decode_alerts
#
# Stop Alerts on experimental TCP options
#
# config disable_tcpopt_experimental_alerts
#
# Stop Alerts on obsolete TCP options
#
# config disable_tcpopt_obsolete_alerts
#
# Stop Alerts on T/TCP alerts
#
# In snort 2.0.1 and above, this only alerts when a TCP option is detected
# that shows T/TCP being actively used on the network.  If this is normal
# behavior for your network, disable the next option.
#
# config disable_tcpopt_ttcp_alerts
#
# Stop Alerts on all other TCPOption type events:
#
# config disable_tcpopt_alerts
#
# Stop Alerts on invalid ip options
#
# config disable_ipopt_alerts
#
# Alert if value in length field (IP, TCP, UDP) is greater than the
# actual length of the captured portion of the packet that the length
# is supposed to represent:
#
# config enable_decode_oversized_alerts
#
# Same as above, but drop packet if in Inline mode -
# enable_decode_oversized_alerts must be enabled for this to work:
#
# config enable_decode_oversized_drops
#

# Configure the detection engine
# ===============================
#
```

```
# Use a different pattern matcher in case you have a machine with very limited
# resources:
#
# config detection: search-method lowmem

# Configure Inline Resets
# ========================
#
# If running an iptables firewall with snort in InlineMode() we can now
# perform resets via a physical device. We grab the indev from iptables
# and use this for the interface on which to send resets. This config
# option takes an argument for the src mac address you want to use in the
# reset packet.  This way the bridge can remain stealthy. If the src mac
# option is not set we use the mac address of the indev device. If we
# don't set this option we will default to sending resets via raw socket,
# which needs an ipaddress to be assigned to the int.
#
# config layer2resets: 00:06:76:DD:5F:E3

##################################################
# Step #2: Configure dynamic loaded libraries
#
# If snort was configured to use dynamically loaded libraries,
# those libraries can be loaded here.
#
# Each of the following configuration options can be done via
# the command line as well.
#
# Load all dynamic preprocessors from the install path
# (same as command line option --dynamic-preprocessor-lib-dir)
#
dynamicpreprocessor directory /usr/local/lib/snort_dynamicpreprocessor/
#
# Load a specific dynamic preprocessor library from the install path
# (same as command line option --dynamic-preprocessor-lib)
#
# dynamicpreprocessor file
/usr/local/lib/snort_dynamicpreprocessor/libdynamicexample.so
#
# Load a dynamic engine from the install path
# (same as command line option --dynamic-engine-lib)
#
dynamicengine /usr/local/lib/snort_dynamicengine/libsf_engine.so
#
# Load all dynamic rules libraries from the install path
# (same as command line option --dynamic-detection-lib-dir)
#
# dynamicdetection directory /usr/local/lib/snort_dynamicrule/
#
# Load a specific dynamic rule library from the install path
# (same as command line option --dynamic-detection-lib)
#
# dynamicdetection file
/usr/local/lib/snort_dynamicrule/libdynamicexamplerule.so
#

##################################################
# Step #3: Configure preprocessors
#
# General configuration for preprocessors is of
# the form
```

```
# preprocessor <name_of_processor>: <configuration_options>

# Configure Flow tracking module
# ------------------------------
#
# The Flow tracking module is meant to start unifying the state keeping
# mechanisms of snort into a single place. Right now, only a portscan detector
# is implemented but in the long term,  many of the stateful subsystems of
# snort will be migrated over to becoming flow plugins. This must be enabled
# for flow-portscan to work correctly.
#
# See README.flow for additional information
#

#preprocessor flow: stats_interval 0 hash 2

# frag3: Target-based IP defragmentation
# --------------------------------------
#
# Frag3 is a brand new IP defragmentation preprocessor that is capable of
# performing "target-based" processing of IP fragments.  Check out the
# README.frag3 file in the doc directory for more background and configuration
# information.
#
# Frag3 configuration is a two step process, a global initialization phase
# followed by the definition of a set of defragmentation engines.
#
# Global configuration defines the number of fragmented packets that Snort can
# track at the same time and gives you options regarding the memory cap for the
# subsystem or, optionally, allows you to preallocate all the memory for the
# entire frag3 system.
#
# frag3_global options:
#   max_frags: Maximum number of frag trackers that may be active at once.
#              Default value is 8192.
#   memcap: Maximum amount of memory that frag3 may access at any given time.
#           Default value is 4MB.
#   prealloc_frags: Maximum number of individual fragments that may be processed
#                   at once.  This is instead of the memcap system, uses static
#                   allocation to increase performance.  No default value.  Each
#                   preallocated fragment typically eats ~1550 bytes.  However,
#                   the exact amount is determined by the snaplen, and this can
#                   go as high as 64K so beware!
#
# Target-based behavior is attached to an engine as a "policy" for handling
# overlaps and retransmissions as enumerated in the Paxson paper.  There are
# currently five policy types available: "BSD", "BSD-right", "First", "Linux"
# and "Last".  Engines can be bound to standard Snort CIDR blocks or
# IP lists.
#
# frag3_engine options:
#   timeout: Amount of time a fragmented packet may be active before expiring.
#            Default value is 60 seconds.
#   ttl_limit: Limit of delta allowable for TTLs of packets in the fragments.
#              Based on the initial received fragment TTL.
#   min_ttl: Minimum acceptable TTL for a fragment, frags with TTLs below this
#            value will be discarded.  Default value is 0.
#   detect_anomalies: Activates frag3's anomaly detection mechanisms.
#   policy: Target-based policy to assign to this engine.  Default is BSD.
#   bind_to: IP address set to bind this engine to.  Default is all hosts.
#
```

```
# Frag3 configuration example:
#preprocessor frag3_global: max_frags 65536, prealloc_frags 65536
#preprocessor frag3_engine: policy linux \
#                           bind_to [10.1.1.12/32,10.1.1.13/32] \
#                           detect_anomalies
#preprocessor frag3_engine: policy first \
#                           bind_to 10.2.1.0/24 \
#                           detect_anomalies
#preprocessor frag3_engine: policy last \
#                           bind_to 10.3.1.0/24
#preprocessor frag3_engine: policy bsd

preprocessor frag3_global: max_frags 65536
preprocessor frag3_engine: policy first detect_anomalies


# stream4: stateful inspection/stream reassembly for Snort
#--------------------------------------------------------------------
# Use in concert with the -z [all|est] command line switch to defeat stick/snot
# against TCP rules.  Also performs full TCP stream reassembly, stateful
# inspection of TCP streams, etc.  Can statefully detect various portscan
# types, fingerprinting, ECN, etc.

# stateful inspection directive
# no arguments loads the defaults (timeout 30, memcap 8388608)
# options (options are comma delimited):
#   detect_scans - stream4 will detect stealth portscans and generate alerts
#                  when it sees them when this option is set
#   detect_state_problems - detect TCP state problems, this tends to be very
#                           noisy because there are a lot of crappy ip stack
#                           implementations out there
#
#   disable_evasion_alerts - turn off the possibly noisy mitigation of
#                            overlapping sequences.
#
#   ttl_limit [number]     - differential of the initial ttl on a session versus
#                             the normal that someone may be playing games.
#                             Routing flap may cause lots of false positives.
#
#   keepstats [machine|binary] - keep session statistics, add "machine" to
#                          get them in a flat format for machine reading, add
#                          "binary" to get them in a unified binary output
#                          format
#   noinspect - turn off stateful inspection only
#   timeout [number] - set the session timeout counter to [number] seconds,
#                       default is 30 seconds
#   max_sessions [number] - limit the number of sessions stream4 keeps
#                          track of
#   memcap [number] - limit stream4 memory usage to [number] bytes (does
#                     not include session tracking, which is set by the
#                     max_sessions option)
#   log_flushed_streams - if an event is detected on a stream this option will
#                          cause all packets that are stored in the stream4
#                          packet buffers to be flushed to disk.  This only
#                          works when logging in pcap mode!
#   server_inspect_limit [bytes] - Byte limit on server side inspection.
#   enable_udp_sessions - turn on tracking of "sessions" over UDP.  Requires
#                          configure --enable-stream4udp.  UDP sessions are
#                          only created when there is a rule for the sender or
#                          responder that has a flow or flowbits keyword.
#   max_udp_sessions [number] - limit the number of simultaneous UDP sessions
```

```
#                                to track
#   udp_ignore_any - Do not inspect UDP packets unless there is a port specific
#                    rule for a given port.  This is a performance improvement
#                    and turns off inspection for udp xxx any -> xxx any rules
#   cache_clean_sessions [number] - Cleanup the session cache by number sessions
#                                   at a time.  The larger the value, the
#                                   more sessions are purged from the cache when
#                                   the session limit or memcap is reached.
#                                   Defaults to 5.
#
#
#
# Stream4 uses Generator ID 111 and uses the following SIDS
# for that GID:
#  SID     Event description
# -----   -------------------
#   1       Stealth activity
#   2       Evasive RST packet
#   3       Evasive TCP packet retransmission
#   4       TCP Window violation
#   5       Data on SYN packet
#   6       Stealth scan: full XMAS
#   7       Stealth scan: SYN-ACK-PSH-URG
#   8       Stealth scan: FIN scan
#   9       Stealth scan: NULL scan
#   10      Stealth scan: NMAP XMAS scan
#   11      Stealth scan: Vecna scan
#   12      Stealth scan: NMAP fingerprint scan stateful detect
#   13      Stealth scan: SYN-FIN scan
#   14      TCP forward overlap

#preprocessor stream4: disable_evasion_alerts

# tcp stream reassembly directive
# no arguments loads the default configuration
#   Only reassemble the client,
#   Only reassemble the default list of ports (See below),
#   Give alerts for "bad" streams
#
# Available options (comma delimited):
#   clientonly - reassemble traffic for the client side of a connection only
#   serveronly - reassemble traffic for the server side of a connection only
#   both - reassemble both sides of a session
#   noalerts - turn off alerts from the stream reassembly stage of stream4
#   ports [list] - use the space separated list of ports in [list], "all"
#                  will turn on reassembly for all ports, "default" will turn
#                  on reassembly for ports 21, 23, 25, 42, 53, 80, 110,
#                  111, 135, 136, 137, 139, 143, 445, 513, 514, 1433, 1521,
#                  2401, and 3306
#   favor_old - favor an old segment (based on sequence number) over a new one.
#               This is the default.
#   favor_new - favor an new segment (based on sequence number) over an old one.
#   overlap_limit [number] - limit on overlaping segments for a session.
#   flush_on_alert - flushes stream when an alert is generated for a session.
#   flush_behavior [mode] -
#           default      - use old static flushpoints (default)
#           large_window - use new larger static flushpoints
#           random       - use random flushpoints defined by flush_base,
#                          flush_seed and flush_range
#   flush_base [number] - lowest allowed random flushpoint (512 by default)
#   flush_range [number] - number is the space within which random flushpoints
```

```
#                           are generated (default 1213)
#   flush_seed [number] - seed for the random number generator, defaults to
#                         Snort PID + time
#
# Using the default random flushpoints, the smallest flushpoint is 512,
# and the largest is 1725 bytes.
#preprocessor stream4_reassemble
#preprocessor stream4_reassemble: both,ports 21 23 25 53 80 110 111 139 143 445
513 1433
# stream5: Target Based stateful inspection/stream reassembly for Snort
# ---------------------------------------------------------------------
# Stream5 is a target-based stream engine for Snort.  Its functionality
# replaces that of Stream4.  Consequently, BOTH Stream4 and Stream5
# cannot be used simultaneously.  Comment out the stream4 configurations
# above to use Stream5.
#
# See README.stream5 for details on the configuration options.
#
# Example config (that emulates Stream4 with UDP support compiled in)
#********************************** da qui
#preprocessor stream5_global: max_tcp 8192, track_tcp yes, \
preprocessor stream5_global: track_tcp yes, \
                             track_udp yes
preprocessor stream5_tcp: policy first, use_static_footprint_sizes
#********************************** fino qui
preprocessor stream5_udp: ignore_any_rules


# Performance Statistics
# ----------------------
# Documentation for this is provided in the Snort Manual.  You should read it.
# It is included in the release distribution as doc/snort_manual.pdf
#
# preprocessor perfmonitor: time 300 file /var/snort/snort.stats pktcnt 10000

# http_inspect: normalize and detect HTTP traffic and protocol anomalies
#
# lots of options available here. See doc/README.http_inspect.
# unicode.map should be wherever your snort.conf lives, or given
# a full path to where snort can find it.
preprocessor http_inspect: global \
    iis_unicode_map unicode.map 1252

preprocessor http_inspect_server: server default \
    profile all ports { 80 8080 8180 } oversize_dir_length 500

#
#  Example unique server configuration
#
#preprocessor http_inspect_server: server 1.1.1.1 \
#    ports { 80 3128 8080 } \
#    server_flow_depth 0 \
#    ascii no \
#    double_decode yes \
#    non_rfc_char { 0x00 } \
#    chunk_length 500000 \
#    non_strict \
#    oversize_dir_length 300 \
#    no_alerts
```

```
# rpc_decode: normalize RPC traffic
# ----------------------------------
# RPC may be sent in alternate encodings besides the usual 4-byte encoding
# that is used by default. This plugin takes the port numbers that RPC
# services are running on as arguments - it is assumed that the given ports
# are actually running this type of service. If not, change the ports or turn
# it off.
# The RPC decode preprocessor uses generator ID 106
#
# arguments: space separated list
# alert_fragments - alert on any rpc fragmented TCP data
# no_alert_multiple_requests - don't alert when >1 rpc query is in a packet
# no_alert_large_fragments - don't alert when the fragmented
#                            sizes exceed the current packet size
# no_alert_incomplete - don't alert when a single segment
#                       exceeds the current packet size

preprocessor rpc_decode: 111 32771

# bo: Back Orifice detector
# -------------------------
# Detects Back Orifice traffic on the network.
#
# arguments:
#   syntax:
#     preprocessor bo: noalert { client | server | general | snort_attack } \
#                      drop    { client | server | general | snort_attack }
#   example:
#     preprocessor bo: noalert { general server } drop { snort_attack }
#
#
# The Back Orifice detector uses Generator ID 105 and uses the
# following SIDS for that GID:
#  SID     Event description
# -----   -------------------
#   1       Back Orifice traffic detected
#   2       Back Orifice Client Traffic Detected
#   3       Back Orifice Server Traffic Detected
#   4       Back Orifice Snort Buffer Attack

preprocessor bo

# ftp_telnet: FTP & Telnet normalizer, protocol enforcement and buff overflow
# ---------------------------------------------------------------------------
# This preprocessor normalizes telnet negotiation strings from telnet and
# ftp traffic.  It looks for traffic that breaks the normal data stream
# of the protocol, replacing it with a normalized representation of that
# traffic so that the "content" pattern matching keyword can work without
# requiring modifications.
#
# It also performs protocol correctness checks for the FTP command channel,
# and identifies open FTP data transfers.
#
# FTPTelnet has numerous options available, please read
# README.ftptelnet for help configuring the options for the global
# telnet, ftp server, and ftp client sections for the protocol.

#####
# Per Step #2, set the following to load the ftptelnet preprocessor
# dynamicpreprocessor file <full path to libsf_ftptelnet_preproc.so>
# or use commandline option
```

```
# --dynamic-preprocessor-lib <full path to libsf_ftptelnet_preproc.so>

preprocessor ftp_telnet: global \
   encrypted_traffic yes \
   inspection_type stateful

preprocessor ftp_telnet_protocol: telnet \
   normalize \
   ayt_attack_thresh 200

# This is consistent with the FTP rules as of 18 Sept 2004.
# CWD can have param length of 200
# MODE has an additional mode of Z (compressed)
# Check for string formats in USER & PASS commands
# Check nDTM commands that set modification time on the file.
preprocessor ftp_telnet_protocol: ftp server default \
   def_max_param_len 100 \
   alt_max_param_len 200 { CWD } \
   cmd_validity MODE < char ASBCZ > \
   cmd_validity MDTM < [ date nnnnnnnnnnnnnn[.n[n[n]]] ] string > \
   chk_str_fmt { USER PASS RNFR RNTO SITE MKD } \
   telnet_cmds yes \
   data_chan

preprocessor ftp_telnet_protocol: ftp client default \
   max_resp_len 256 \
   bounce yes \
   telnet_cmds yes

# smtp: SMTP normalizer, protocol enforcement and buffer overflow
# ----------------------------------------------------------------------------
# This preprocessor normalizes SMTP commands by removing extraneous spaces.
# It looks for overly long command lines, response lines, and data header lines.
# It can alert on invalid commands, or specific valid commands.  It can
optionally
# ignore mail data, and can ignore TLS encrypted data.
#
# SMTP has numerous options available, please read README.SMTP for help
# configuring options.

#####
# Per Step #2, set the following to load the smtp preprocessor
# dynamicpreprocessor file <full path to libsf_smtp_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_smtp_preproc.so>

preprocessor smtp: \
  ports { 25 587 691 } \
  inspection_type stateful \
  normalize cmds \
  normalize_cmds { EXPN VRFY RCPT } \
  alt_max_command_line_len 260 { MAIL } \
  alt_max_command_line_len 300 { RCPT } \
  alt_max_command_line_len 500 { HELP HELO ETRN } \
  alt_max_command_line_len 255 { EXPN VRFY }

# sfPortscan
# ----------
# Portscan detection module.  Detects various types of portscans and
# portsweeps.  For more information on detection philosophy, alert types,
# and detailed portscan information, please refer to the README.sfportscan.
```

```
#
# -configuration options-
#     proto { tcp udp icmp ip all }
#       The arguments to the proto option are the types of protocol scans that
#       the user wants to detect.  Arguments should be separated by spaces and
#       not commas.
#     scan_type { portscan portsweep decoy_portscan distributed_portscan all }
#       The arguments to the scan_type option are the scan types that the
#       user wants to detect.  Arguments should be separated by spaces and not
#       commas.
#     sense_level { low|medium|high }
#       There is only one argument to this option and it is the level of
#       sensitivity in which to detect portscans.  The 'low' sensitivity
#       detects scans by the common method of looking for response errors, such
#       as TCP RSTs or ICMP unreachables.  This level requires the least
#       tuning.  The 'medium' sensitivity level detects portscans and
#       filtered portscans (portscans that receive no response).  This
#       sensitivity level usually requires tuning out scan events from NATed
#       IPs, DNS cache servers, etc.  The 'high' sensitivity level has
#       lower thresholds for portscan detection and a longer time window than
#       the 'medium' sensitivity level.  Requires more tuning and may be noisy
#       on very active networks.  However, this sensitivity levels catches the
#       most scans.
#     memcap { positive integer }
#       The maximum number of bytes to allocate for portscan detection.  The
#       higher this number the more nodes that can be tracked.
#     logfile { filename }
#       This option specifies the file to log portscan and detailed portscan
#       values to.  If there is not a leading /, then snort logs to the
#       configured log directory.  Refer to README.sfportscan for details on
#       the logged values in the logfile.
#     watch_ip { Snort IP List }
#     ignore_scanners { Snort IP List }
#     ignore_scanned { Snort IP List }
#       These options take a snort IP list as the argument.  The 'watch_ip'
#       option specifies the IP(s) to watch for portscan.  The
#       'ignore_scanners' option specifies the IP(s) to ignore as scanners.
#       Note that these hosts are still watched as scanned hosts.  The
#       'ignore_scanners' option is used to tune alerts from very active
#       hosts such as NAT, nessus hosts, etc.  The 'ignore_scanned' option
#       specifies the IP(s) to ignore as scanned hosts.  Note that these hosts
#       are still watched as scanner hosts.  The 'ignore_scanned' option is
#       used to tune alerts from very active hosts such as syslog servers, etc.
#     detect_ack_scans
#       This option will include sessions picked up in midstream by the stream
#       module, which is necessary to detect ACK scans.  However, this can lead
to
#       false alerts, especially under heavy load with dropped packets; which is
why
#       the option is off by default.
#
preprocessor sfportscan: proto  { all } \
                         memcap { 10000000 } \
                  scan_type { all } \
                         sense_level { low }


# arpspoof
#--------------------------------------
# Experimental ARP detection code from Jeff Nathan, detects ARP attacks,
# unicast ARP requests, and specific ARP mapping monitoring.  To make use of
```

```
# this preprocessor you must specify the IP and hardware address of hosts on
# the same layer 2 segment as you.  Specify one host IP MAC combo per line.
# Also takes a "-unicast" option to turn on unicast ARP request detection.
00:19:99:31:7A:C8
# Arpspoof uses Generator ID 112 and uses the following SIDS for that GID:

#  SID     Event description
# -----   -------------------
#   1       Unicast ARP request
#   2       Etherframe ARP mismatch (src)
#   3       Etherframe ARP mismatch (dst)
#   4       ARP cache overwrite attack

preprocessor arpspoof
#preprocessor arpspoof_detect_host: 192.168.40.1 f0:0f:00:f0:0f:00
preprocessor arpspoof_detect_host: 172.16.252.1 00:19:99:31:7A:C8
# ssh
#--------------------------------------
# EXPERIMENTAL CODE!!!
#
# THIS CODE IS STILL EXPERIMENTAL AND MAY OR MAY NOT BE STABLE!
# USE AT YOUR OWN RISK!  DO NOT USE IN PRODUCTION ENVIRONMENTS.
# YOU HAVE BEEN WARNED.
#
# The SSH preprocessor detects the following exploits: Gobbles, CRC 32,
# Secure CRT, and the Protocol Mismatch exploit.
#
# Both Gobbles and CRC 32 attacks occur after the key exchange, and are
# therefore encrypted.  Both attacks involve sending a large payload
# (20kb+) to the server immediately after the authentication challenge.
# To detect the attacks, the SSH preprocessor counts the number of bytes
# transmitted to the server.  If those bytes exceed a pre-defined limit
# within a pre-define number of packets, an alert is generated.  Since
# Gobbles only effects SSHv2 and CRC 32 only effects SSHv1, the SSH
# version string exchange is used to distinguish the attacks.
#
# The Secure CRT and protocol mismatch exploits are observable before
# the key exchange.
#
# SSH has numerous options available, please read README.ssh for help
# configuring options.

#####
# Per Step #2, set the following to load the ssh preprocessor
# dynamicpreprocessor file <full path to libsf_ssh_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_ssh_preproc.so>
#
#preprocessor ssh: server_ports { 22 } \
#                  max_client_bytes 19600 \
#                  max_encrypted_packets 20

# DCE/RPC
#--------------------------------------
#
# The dcerpc preprocessor detects and decodes SMB and DCE/RPC traffic.
# It is primarily interested in DCE/RPC data, and only decodes SMB
# to get at the DCE/RPC data carried by the SMB layer.
#
# Currently, the preprocessor only handles reassembly of fragmentation
# at both the SMB and DCE/RPC layer.  Snort rules can be evaded by
```

```
# using both types of fragmentation; with the preprocessor enabled
# the rules are given a buffer with a reassembled SMB or DCE/RPC
# packet to examine.
#
# At the SMB layer, only fragmentation using WriteAndX is currently
# reassembled.  Other methods will be handled in future versions of
# the preprocessor.
#
# Autodetection of SMB is done by looking for "\xFFSMB" at the start of
# the SMB data, as well as checking the NetBIOS header (which is always
# present for SMB) for the type "SMB Session".
#
# Autodetection of DCE/RPC is not as reliable.  Currently, two bytes are
# checked in the packet.  Assuming that the data is a DCE/RPC header,
# one byte is checked for DCE/RPC version (5) and another for the type
# "DCE/RPC Request".  If both match, the preprocessor proceeds with that
# assumption that it is looking at DCE/RPC data.  If subsequent checks
# are nonsensical, it ends processing.
#
# DCERPC has numerous options available, please read README.dcerpc for help
# configuring options.

#####
# Per Step #2, set the following to load the dcerpc preprocessor
# dynamicpreprocessor file <full path to libsf_dcerpc_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_dcerpc_preproc.so>

preprocessor dcerpc: \
    autodetect \
    max_frag_size 3000 \
    memcap 100000

# DNS
#--------------------------------------
# The dns preprocessor (currently) decodes DNS Response traffic
# and detects a few vulnerabilities.
#
# DNS has a few options available, please read README.dns for
# help configuring options.

#####
# Per Step #2, set the following to load the dns preprocessor
# dynamicpreprocessor file <full path to libsf_dns_preproc.so>
# or use commandline option
# --dynamic-preprocessor-lib <full path to libsf_dns_preproc.so>

preprocessor dns: \
    ports { 53 } \
    enable_rdata_overflow

# SSL
#--------------------------------------
# Encrypted traffic should be ignored by Snort for both performance reasons
# and to reduce false positives.  The SSL Dynamic Preprocessor (SSLPP)
# inspects SSL traffic and optionally determines if and when to stop
# inspection of it.
#
# Typically, SSL is used over port 443 as HTTPS.  By enabling the SSLPP to
# inspect port 443, only the SSL handshake of each connection will be
# inspected.  Once the traffic is determined to be encrypted, no further
```

```
# inspection of the data on the connection is made.
#
#   Important note: Stream4 or Stream5 should be explicitly told to reassemble
#                   traffic on the ports that you intend to inspect SSL
#                   encrypted traffic on.
#
#   To add reassembly on port 443 to Stream5, use 'port both 443' in the
#   Stream5 configuration.

preprocessor ssl: noinspect_encrypted


##################################################################
# Step #4: Configure output plugins
#
# Uncomment and configure the output plugins you decide to use.  General
# configuration for output plugins is of the form:
#
# output <name_of_plugin>: <configuration_options>
#
# alert_syslog: log alerts to syslog
# ----------------------------------
# Use one or more syslog facilities as arguments.  Win32 can also optionally
# specify a particular hostname/port.  Under Win32, the default hostname is
# '127.0.0.1', and the default port is 514.
#
# [Unix flavours should use this format...]
# output alert_syslog: LOG_AUTH LOG_ALERT
#
# [Win32 can use any of these formats...]
# output alert_syslog: LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname, LOG_AUTH LOG_ALERT
# output alert_syslog: host=hostname:port, LOG_AUTH LOG_ALERT

# log_tcpdump: log packets in binary tcpdump format
# -------------------------------------------------
# The only argument is the output file name.
#
# output log_tcpdump: tcpdump.log

# database: log to a variety of databases
# ---------------------------------------
# See the README.database file for more information about configuring
# and using this plugin.
#
output database: log, mysql, user=snort password=snortdb dbname=snort
host=localhost
output database: alert, mysql, user=snort password=snortdb dbname=snort
host=localhost
# output database: alert, postgresql, user=snort dbname=snort
# output database: log, odbc, user=snort dbname=snort
# output database: log, mssql, dbname=snort user=snort password=test
# output database: log, oracle, dbname=snort user=snort password=test

# unified: Snort unified binary format alerting and logging
# ---------------------------------------------------------
# The unified output plugin provides two new formats for logging and generating
# alerts from Snort, the "unified" format.  The unified format is a straight
# binary format for logging data out of Snort that is designed to be fast and
# efficient.  Used with barnyard (the new alert/log processor), most of the
# overhead for logging and alerting to various slow storage mechanisms such as
```

```
# databases or the network can now be avoided.
#
# Check out the spo_unified.h file for the data formats.
#
# Two arguments are supported.
#    filename - base filename to write to (current time_t is appended)
#    limit    - maximum size of spool file in MB (default: 128)
#
# output alert_unified: filename snort.alert, limit 128
# output log_unified: filename snort.log, limit 128


# prelude: log to the Prelude Hybrid IDS system
# ------------------------------------------------
#
# profile = Name of the Prelude profile to use (default is snort).
#
# Snort priority to IDMEF severity mappings:
# high < medium < low < info
#
# These are the default mapped from classification.config:
# info   = 4
# low    = 3
# medium = 2
# high   = anything below medium
#
# output alert_prelude
# output alert_prelude: profile=snort-profile-name


# You can optionally define new rule types and associate one or more output
# plugins specifically to that type.
#
# This example will create a type that will log to just tcpdump.
# ruletype suspicious
# {
#   type log
#   output log_tcpdump: suspicious.log
# }
#
# EXAMPLE RULE FOR SUSPICIOUS RULETYPE:
# suspicious tcp $HOME_NET any -> $HOME_NET 6667 (msg:"Internal IRC Server";)
#
# This example will create a rule type that will log to syslog and a mysql
# database:
# ruletype redalert
# {
#   type alert
#   output alert_syslog: LOG_AUTH LOG_ALERT
#   output database: log, mysql, user=snort dbname=snort host=localhost
# }
#
# EXAMPLE RULE FOR REDALERT RULETYPE:
# redalert tcp $HOME_NET any -> $EXTERNAL_NET 31337 \
#   (msg:"Someone is being LEET"; flags:A+;)

#
# Include classification & priority settings
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\classification.config
#
```

```
include classification.config

#
# Include reference systems
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\reference.config
#

include reference.config

##################################################################
# Step #5: Configure snort with config statements
#
# See the snort manual for a full set of configuration references
#
# config flowbits_size: 64
#
# New global ignore_ports config option from Andy Mullican
#
# config ignore_ports: <tcp|udp> <list of ports separated by whitespace>
# config ignore_ports: tcp 21 6667:6671 1356
# config ignore_ports: udp 1:17 53


##################################################################
# Step #6: Customize your rule set
#
# Up to date snort rules are available at http://www.snort.org
#
# The snort web site has documentation about how to write your own custom snort
# rules.

#========================================
# Include all relevant rulesets here
#
# The following rulesets are disabled by default:
#
#   web-attacks, backdoor, shellcode, policy, porn, info, icmp-info, virus,
#   chat, multimedia, and p2p
#
# These rules are either site policy specific or require tuning in order to not
# generate false positive alerts in most enviornments.
#
# Please read the specific include file for more information and
# README.alert_order for how rule ordering affects how alerts are triggered.
#========================================

include $RULE_PATH/local.rules
include $RULE_PATH/bad-traffic.rules
include $RULE_PATH/exploit.rules
include $RULE_PATH/scan.rules
include $RULE_PATH/finger.rules
include $RULE_PATH/ftp.rules
include $RULE_PATH/telnet.rules
include $RULE_PATH/rpc.rules
include $RULE_PATH/rservices.rules
include $RULE_PATH/dos.rules
include $RULE_PATH/ddos.rules
include $RULE_PATH/dns.rules
include $RULE_PATH/tftp.rules
```

```
include $RULE_PATH/web-cgi.rules
include $RULE_PATH/web-coldfusion.rules
include $RULE_PATH/web-iis.rules
include $RULE_PATH/web-frontpage.rules
include $RULE_PATH/web-misc.rules
include $RULE_PATH/web-client.rules
include $RULE_PATH/web-php.rules

include $RULE_PATH/sql.rules
include $RULE_PATH/x11.rules
include $RULE_PATH/icmp.rules
include $RULE_PATH/netbios.rules
include $RULE_PATH/misc.rules
include $RULE_PATH/attack-responses.rules
include $RULE_PATH/oracle.rules
include $RULE_PATH/mysql.rules
include $RULE_PATH/snmp.rules

include $RULE_PATH/smtp.rules
include $RULE_PATH/imap.rules
include $RULE_PATH/pop2.rules
include $RULE_PATH/pop3.rules

include $RULE_PATH/nntp.rules
include $RULE_PATH/other-ids.rules
include $RULE_PATH/web-attacks.rules
# include $RULE_PATH/backdoor.rules
include $RULE_PATH/shellcode.rules
# include $RULE_PATH/policy.rules
# include $RULE_PATH/porn.rules
# include $RULE_PATH/info.rules
include $RULE_PATH/icmp-info.rules
# include $RULE_PATH/virus.rules
# include $RULE_PATH/chat.rules
# include $RULE_PATH/multimedia.rules
# include $RULE_PATH/p2p.rules
# include $RULE_PATH/spyware-put.rules
# include $RULE_PATH/specific-threats.rules
include $RULE_PATH/experimental.rules

# include $PREPROC_RULE_PATH/preprocessor.rules
# include $PREPROC_RULE_PATH/decoder.rules

# Include any thresholding or suppression commands. See threshold.conf in the
# <snort src>/etc directory for details. Commands don't necessarily need to be
# contained in this conf, but a separate conf makes it easier to maintain them.
# Note for Windows users:  You are advised to make this an absolute path,
# such as:  c:\snort\etc\threshold.conf
# Uncomment if needed.
# include threshold.conf
```

Bibliografia

Snort.org,

http://www.snort.org/

Open Skills, "*Installazione Snort"*

http://openskill.info/infobox.php?ID=585

Open Skills, "*Intrusion Detection System (IDS) su Linux"*

http://openskill.info/infobox.php?ID=644

Open Skills, "*Configurare Named per l'aggiornamento dinamico mediante DHCP"*

http://openskill.info/infobox.php?IDbox=766

Open Skills, "*Basi di SQL su MySQL"*

http://openskill.info/topic.php?ID=163

Open Skills, "*Installazione e configurazione di MySQL"*

http://openskill.info/topic.php?ID=152

Open Skills, "Utilizzo PEAR*"*

http://openskill.info/topic.php?ID=1437

FWBuilder,

http://www.fwbuilder.org/

PEAR,

http://pear.php.net/

Wikibooks,"Snort"

http://it.wikibooks.org/wiki/Snort

NixCraft, *"Install Squid Proxy Server on CentOS / Redhat enterprise Linux 5"*

http://www.cyberciti.biz/tips/howto-rhel-centos-fedora-squid-installation-configuration.html

IlSoftware.it, *"Installazione, configurazione..."*

http://www.ilsoftware.it/articoli.asp?id=2181

IlSoftware.it, *"Installazione, configurazione..."*

http://www.ilsoftware.it/articoli.asp?id=2181&pag=5

Webmin.com, "*Webmin"*

http://www.webmin.com/

Server.html.it, "*Webmin: Gestire un server da remoto*"

http://server.html.it/articoli/leggi/2999/webmin-gestire-un-server-da-remoto/

Linuxpedia, "*Webmin*"

http://linuxpedia.netsons.org/index.php?title=Webmin

Andrew S. Tanenbaum, "Reti di calcolatori" ,

4 edizione, Pearson

Andrew Lockhart, "Sicurezza delle reti trucchi e segreti"

2 Edizione, Tecniche Nuove Edizioni