UNIVERSITÀ DEGLI STUDI DI PAVIA
FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA IN INGEGNERIA INFORMATICA

IMPLEMENTAZIONE DI UN SISTEMA DI MONITORAGGIO DI RETE L.A.N. CON TOOLS "OPEN SOURCE"

Tutore Aziendale: Mauro Bruseghini
Tutore Universitario: Francesco Leporati

Tesi di laurea di
Matteo Vitolo

ANNO ACCADEMICO 2002-2003

SOMMARIO

Capitolo 1

INTRODUZIONE

Negli ultimi anni si è assistito ad una notevole evoluzione dei servizi offerti dalle reti aziendali.

Oggi è comune trovare, anche in piccole aziende, server per la gestione della posta elettronica, del sito web aziendale, del sito internet oltre ai tradizionali file server.

Quindi si può affermare che ci sono servizi di cui un'azienda moderna non può fare a meno. Il compito di un buon sistema di monitoraggio è controllare che questi servizi siano sempre attivi e raggiungibili.

Un sito internet momentaneamente non raggiungibile può provocare danni a livello d'immagine, e notevoli perdite economiche per l'azienda. Perciò tutti i servizi che si affacciano su internet e che sono accessibili dall'esterno devono essere continuamente monitorati a intervalli prestabiliti.

Inoltre un sistema di monitoraggio deve controllare le risorse dei terminali di maggior importanza all'interno della rete aziendale. Se il terminale adibito a web-server ha troppi processi attivi questo potrebbe rallentare la macchina e quindi non fornire in modo adeguato il proprio servizio.

Naturalmente è di primaria importanza l'implementazione del sistema di monitoraggio in una situazione di piena sicurezza; di conseguenza la macchina di management sarà posta dietro ad un firewall, i dati che fluiscono attraverso la rete devono essere crittografati e chi ha accesso ai dati del monitoraggio ci può arrivare attraverso le opportune autenticazioni.

Un sistema di monitoraggio efficiente e completo è solo uno specchio della situazione del cliente: informa chi è adibito all'assistenza di quali servizi o terminali sono attivi o disattivi e non prende nessuna decisione attiva su come risolvere i problemi.
Sarà compito di chi è adibito all'assistenza risolvere il problema proposto dal sistema di monitoraggio.

Molti sono i sistemi di monitoraggio di reti presenti sul mercato e hanno caratteristiche in linea di massima pressoché similari.
Il prodotto leader dei sistemi di monitoraggio, il più usato e sicuramente il più famoso è della Hewlett Packard ed è chiamato OpenView.

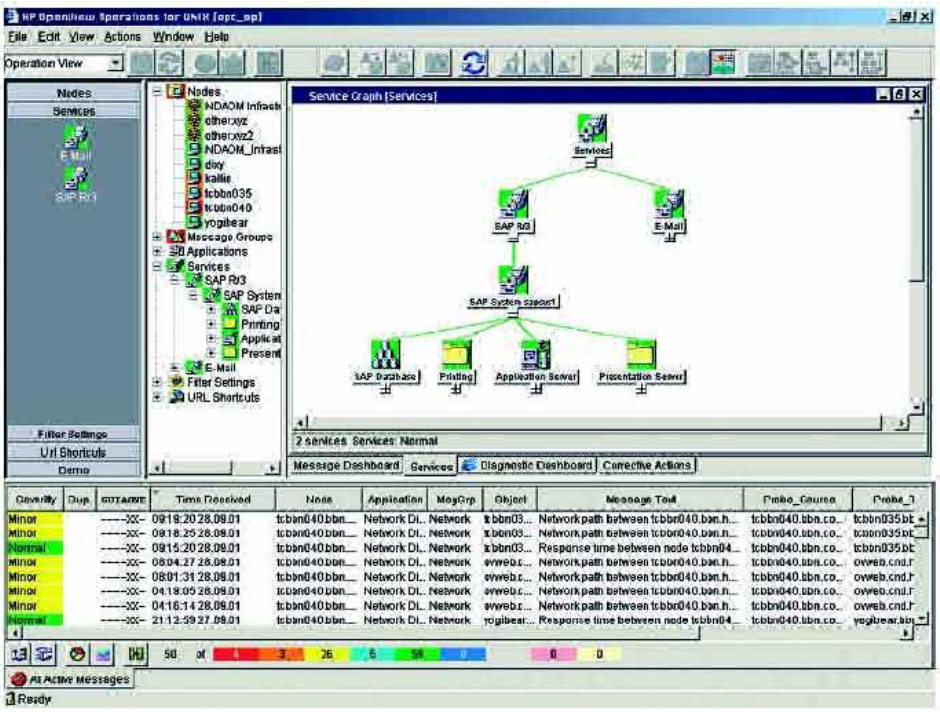

Fig. 1 - HP OpenView: Interfaccia utente

OpenView può essere composto da più di ventisette componenti tra cui Network Node Manager (NNM) 6.31, Performance Manager and

Performance Insight 4.5, Operations 7.0, Storage Area Manager, Internet Services 4.0 e OpenView Reporter.

Il cuore del programma di OpenView è Network Node Manager che, come sottolinea la casa costruttrice, è in grado di configurare automaticamente la rete da monitorare.

Purtroppo, data la grande complessità su cui si basa un modulo di auto-rilevamento, è facile che il programma non venga pienamente compreso e utilizzato nella sua completa potenzialità. Molti utenti di OpenView si trovano, infatti, costretti a riconfigurare manualmente la rete auto-rilevata dal programma perché non conforme alla realtà dei fatti vanificando quindi lo scopo primario del modulo.

Il nucleo di OpenView è basato sul protocollo SNMP acronimo di Simple Network Management Protocol; la macchina su cui gira il client SNMP crea "trap" che vengono inviati dalla macchina di management sui cui gira OpenView che poi genererà gli eventuali "alert" da spedire al cliente.

Si rende quindi necessario l'acquisto dell'apposito modulo per il monitoraggio dei servizi; questo modulo aggiuntivo controlla tutti i servizi che si affacciano sulla rete.

Il grande numero di moduli aggiuntivi rendono OpenView un sistema di monitoraggio molto personalizzabile.

Importante è, come in tutti i sistemi di monitoraggio, l'invio delle notifiche che possono avvenire via mail o con la creazione di report giornalieri in formato HTML.

Uno dei punti di debolezza di questo programma è la mancanza di un'interfaccia sul web. In pratica, chi non è sul terminale che funge da macchina di management non ha la possibilità di avere un riscontro in tempo reale sull'effettivo stato di salute della rete se

non con le classiche notifiche via e-mail che avvisano l'amministratore di rete quando il danno è già attivo sulla lan.
Il costo già elevato del programma viene evidenziato poi dall'aggiunta dei moduli facoltativi ma indispensabili per il buon funzionamento del sistema di monitoraggio. Inoltre c'è da aggiungere il costo di implementazione e installazione del programma, nonché il costo di manutenzione e infine il costo di aggiornamento di chi sarà adibito all'utilizzo del programma.
HP OpenView è caratterizzato da una certa difficoltà d'uso. Inoltre la documentazione sul web sembra essere non molto esauriente a scapito della facilità di utilizzo.

Concorrente principale di OpenView della Hewlett Packard è un prodotto della IBM denominato Tivoli-netview.
Tivoli è costituito da un insieme di programmi adibiti al controllo completo di tutti gli aspetti di una rete aziendale.
Il nucleo è formato da un modulo principale chiamato Tivoli Management Platform che è in pratica un interfacciamento al database in cui vengono salvati tutti i risultati ottenuti. La banca dati e il TMP sono la base del programma senza i quali i moduli da soli, come netview, non possono funzionare.
Non solo, il programma Tivoli è un sistema che funziona esclusivamente su AIX che è il sistema operativo, basato su UNIX, di IBM che naturalmente può lavorare esclusivamente su macchine RISC IBM.
Tutto questo dà come risultato un costo elevato che grava sull'economia dell'azienda.
Netview è comunque basato su un modulo di auto-rilevamento e auto-configurazione dei nodi. Il modulo, che viene denominato

Netmon, scopre i nodi IP nella rete a scadenza di un intervallo di tempo che può essere configurato. Questo modulo è basato sulla presenza di un agent SNMP.
Questo dà la possibilità di utilizzare i dati che provengono dagli agent SNMP in tempo reale e di proporli sotto forma di grafico o tabella all'utente.
Inoltre Tivoli Netview utilizza un motore che costruisce graficamente guide per diagnosticare i problemi all'amministratore senza segnalare tutti i sintomi. Netview può inoltre gestire un numero di azioni di risposta a eventi e eccezioni e riferire all'amministratore via e-mail o SMS via cellulare.
Infine, è possibile, con il modulo denominato Tivoli Netview Web Console, controllare l'effettivo stato di salute da differenti postazioni connettendosi ad un sito web creato automaticamente da questo servizio che riporta in tempo reale i dati prodotti dal sistema di monitoraggio.
Importante è sottolineare la possibilità di creare un sistema di monitoraggio distribuito che dia l'opportunità di monitorare ciò che è posto dietro un "firewall" e quindi inaccessibile dall'esterno.

Fino a questo momento sono stati presi in considerazioni due prodotti di due case di software, la Hewlett-Packard e la IBM, che vendono i loro programmi senza pubblicare il codice sorgente. Questo vuol dire: gli sviluppatori del programma sono solo e rimarranno solamente i programmatori HP nel caso di OpenView e i programmatori IBM nel caso di Netview.
Negli ultimi anni è enormemente cresciuta la filosofia dell'Open Source. Il concetto di fondo che sta dietro al software libero è che, pubblicando in rete tutti i codici sorgenti, chiunque abbia un minimo

di conoscenza del linguaggio con cui sono scritti può leggerli, modificarli per il loro uso e migliorarli.

Per quanto gli sviluppatori della IBM e della HP possano essere maghi della programmazione sono tuttavia un numero finito. Invece un programma Open Source ha un contributo continuo di sviluppatori a livello globale, ci lavorano costantemente programmatori da tutto il mondo apportando tutti una diversa esperienza con una passione e una costanza a volte diversa da chi lo fa per lavoro: quindi un software non libero può non competere con il continuo aggiornamento che un programma che pubblica i propri codici sorgenti su internet può avere.

In questa ottica possiamo prendere in considerazione NetSaint che è stato uno dei migliori software di monitoraggio di reti a livello "Open Source". Uso il tempo passato prossimo perché non lo è più.

Ethan Galstad vero e proprio padre di questo programma, insieme ad un buon numero sviluppatori ufficiali, nel 2001 hanno deciso di chiudere per sempre il progetto NetSaint. La motivazione, molto fumosa a dir la verità, si basa sul mal funzionamento di alcuni moduli del programma.

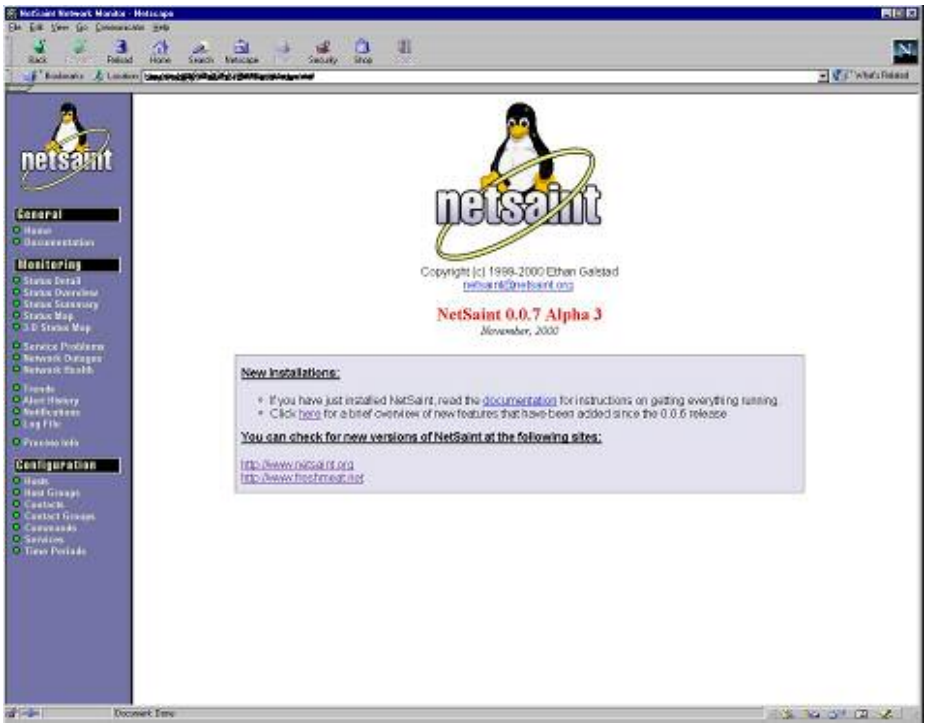

Fig. 2 – Interfaccia web di netsaint

Dalle ceneri di NetSaint, grazie allo stesso Ethan Galstad, insieme a buona parte degli sviluppatori ufficiali di NetSaint, è nato il programma Nagios, con un nucleo centrale completamente rinnovato ma che al vecchio utente di NetSaint non sembra molto differente.

Nagios, dice sempre Ethan Galstad, è l'acronimo di "Nagios Ain't Gonna Insist On Sainthood" che appunto spiega come questo programma sia il successore di NetSaint ma non ne voglia più sentire parlare (trad: Nagios non insisterà più sui santi).

A parte gli scherzi, un altro acronimo che il padre di Nagios ci suggerisce per descrivere in un modo più appropriato il suo programma, è "Notices Any Glitch In Our System" che sta a significare "Notifichiamo qualsiasi piccolo problema sul nostro sistema".

Perché utilizzare questo software? Innanzitutto Nagios produce gli stessi identici risultati di OpenView e Tivoli Netview solo che questi sono molto più ostici da utilizzare e in più presentano dei costi elevati, considerando programma, installazione, implementazione, manutenzione e gestione.

Nagios essendo un software libero ha come unici costi, peraltro esigui, quelli di installazione e implementazione. Inoltre non essendo un programma complesso come gli altri due, è molto più semplice da utilizzare e da comprendere.

Così la scelta è ricaduta su Nagios per un fattore puramente redditizio: se non fosse stato conveniente e se non avesse fornito un servizio almeno equivalente a quello dei software a pagamento ci si sarebbe orientati su OpenView o Netview.

Quindi in questo documento sarà spiegato quali programmi e quali protocolli sono stati utilizzati e come questi sono stati implementati in modo tale da funzionare al meglio nei capitoli 2 e 3. In più è importante che il programma rispetti certi canoni di sicurezza che non si possono trascurare quando si parla di reti che sono descritti nel capitolo 4. Infine nell'ultimo capitolo sono presenti le conclusioni di come si è svolto il tirocinio e sull'implementazione del sistema di monitoraggio.

Capitolo 2

SCHEMA GENERALE

2.1 Software utilizzato

Nagios è un programma di monitoraggio di reti che è stato concepito e sviluppato per lavorare in ambiente Linux e in generale in presenza di tutti i possibili sistemi Unix. L'unico requisito di base che Nagios ha è la presenza sulla macchina di un compilatore C. Questo servirà per compilare i file sorgente e generare sia gli eseguibili sia il programma vero e proprio.
In realtà il discorso è un pochino più complesso. Scaricando l'ultima "release" di Nagios si avrà solamente il nucleo del programma che è in grado di far eseguire i "plugin", mandare le notifiche e generare l'interfaccia web. Ma non è in grado di controllare nessun servizio.
In realtà è un altro il modulo adibito a fare i controlli, i cosiddetti "check". Infatti, senza i "plugin", scaricabili dallo stesso sito da cui si preleva Nagios, il programma non è in grado di controllare nessun servizio.
Per quanto riguarda le richieste per la generazione dell'interfaccia http, Nagios necessita la presenza di un HTTP server sul terminale adibito a console di monitoraggio, nella fattispecie Apache, e una libreria denominata "gd library" per la creazione di grafici e disegni.

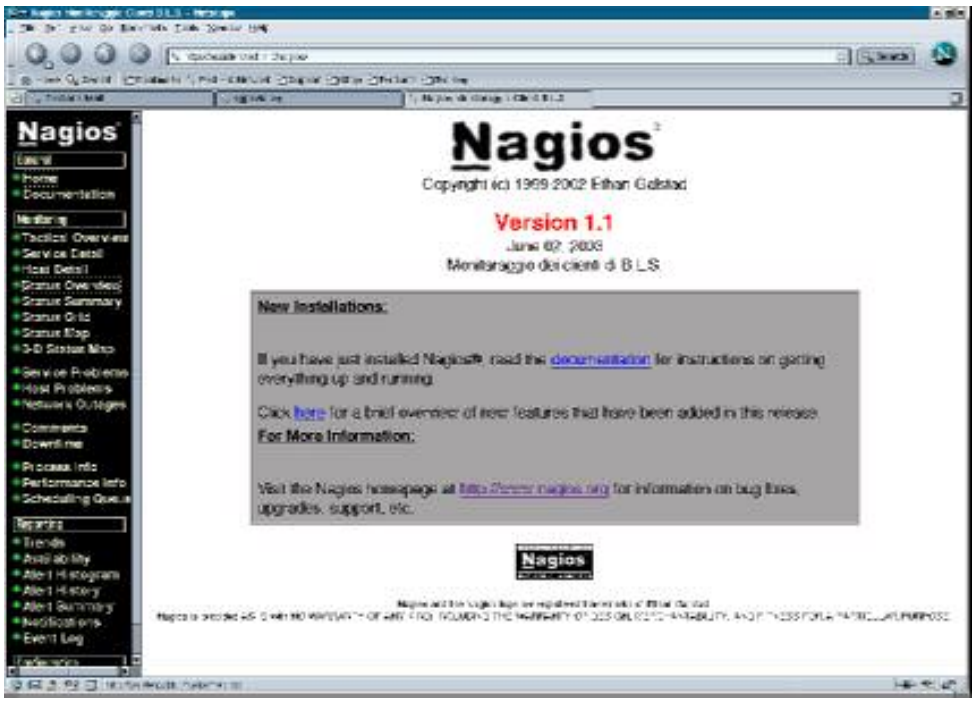

Fig. 3 – Prima pagina dell'interfaccia Web di Nagios

Apache è un programma che dà la possibilità a chi si connette ad un indirizzo internet di poter consultare il sito che risiede sulla macchina con un qualsiasi web browser, sia questo Internet Explorer o Netscape.

Inoltre è il più importante e il più utilizzato web server al mondo, basti dire che più del 60% dei server web lo utilizzano.

Apache è il successore di httpd che era il più importante web server prima del 1995, sviluppato da Rob McCool per la National Center for Supercomputing Applications (NCSA), presso la University of Illinois. Quando, nel 1994, Rob McCool esce dal NCSA il progetto httpd non prosegue ulteriormente. Dal 1995 un gruppo di webmaster, orfani del loro web server preferito, riprende in mano le sorti di httpd e fanno nascere Apache Http server, migliorandolo e ampliando il progetto con una serie di moduli paralleli.

La GD Graphics Library, invece, è una libreria ANSI C per la creazione dinamica di immagini PNG e JPEG utilizzata soprattutto

con Apache. Questa libreria disegna automaticamente immagini complete di linee, archi, testi, porzioni di altre immagini, in formato PNG e JPEG, che sono i formati più utilizzati nelle applicazioni del World Wide Web.

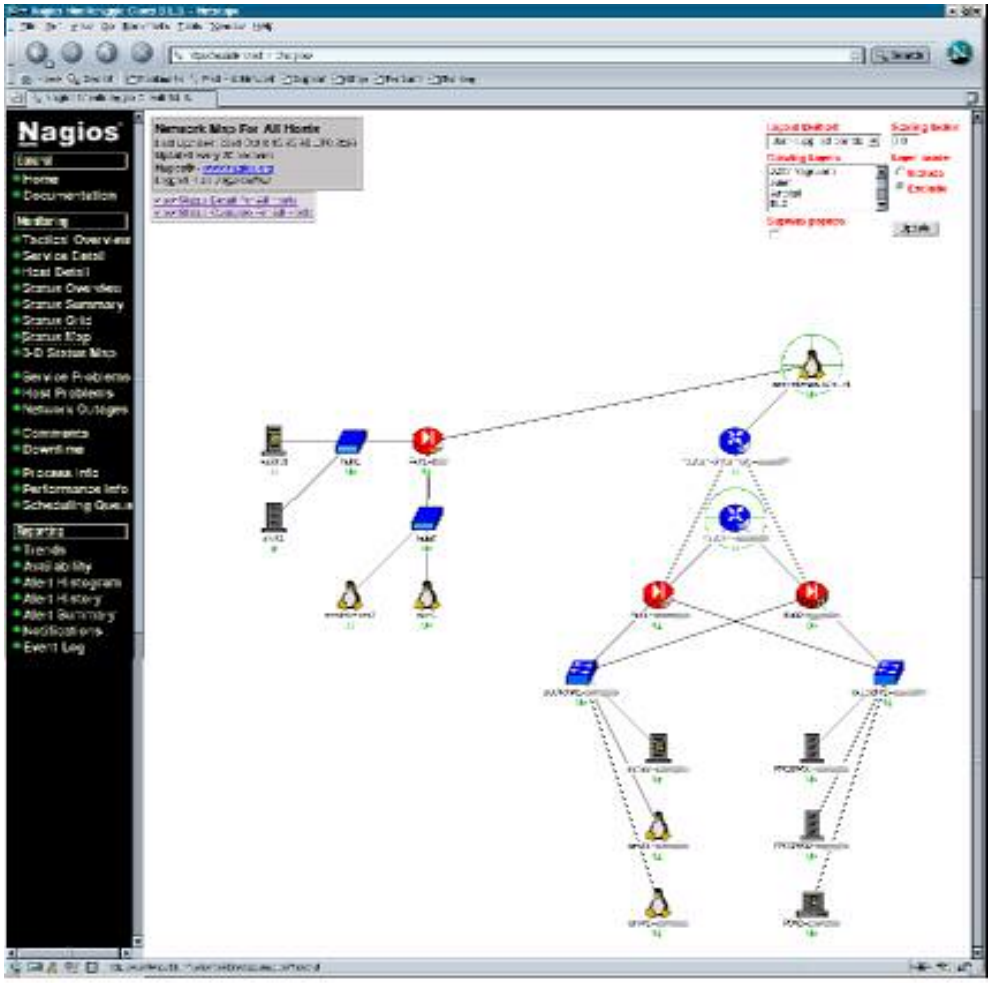

Fig. 4 – Una mappa generata automaticamente da Nagios

Questa libreria è utilizzata da Nagios per creare i grafici riguardanti la raggiungibilità di servizi e host e per disegnare le mappe che descrivono la rete.

Ritornando alla configurazione base, dopo avere compilato il nucleo di Nagios verrà creata una cartella */usr/local/nagios* con all'interno altre sei cartelle:

- *bin* dove risiedono gli eseguibili di Nagios,
- *etc* dove possiamo trovare tutti i file di configurazione,
- *libexec* dove risiedono tutti i "plugin",
- *sbin* dove sono tutti i file .cgi che generano alcune pagine dinamiche per l'interfaccia web,
- *share* dove sono tutti i file statici che servono per la generazione di pagine HTML
- *var* dove vengono salvati tutti i file di log che Nagios produce

I file che sono nella cartella libexec sono ciò che gli sviluppatori di Nagios chiamano "plugin". Compilando il file sorgente dei plugin verranno creati in questa cartella un buon numero di eseguibili. I "plugin" sono eseguibili o script, scritti in C, Perl o qualsiasi altro linguaggio di programmazione, che possono essere eseguiti da una linea di comando di una shell di linux per controllare lo stato di un host o di un servizio.

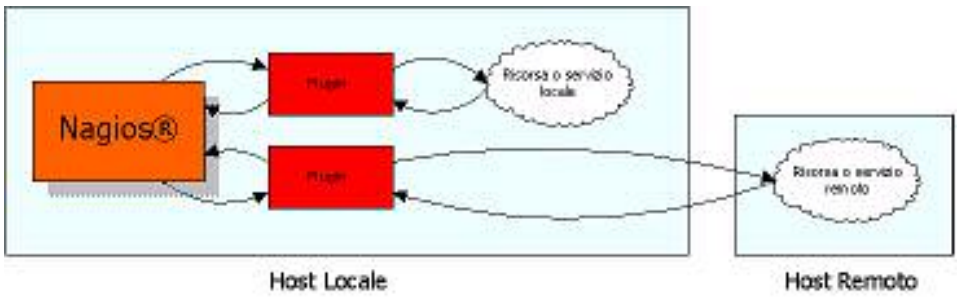

Fig. 5 – Funzionamento dei "plugin" di Nagios

Nagios utilizzerà il risultato ottenuto dall'esecuzione di un "plugin" per determinare l'attuale stato di salute degli host o dei servizi sulla rete. La forza di Nagios sta proprio in questo: se nasce la necessità di controllare un nuovo servizio bisogna solamente scrivere l'apposito script capace di controllare la nuova risorsa e aggiungere le necessarie configurazioni. In pratica c'è una completa

personalizzazione del programma e tutti i servizi possibili ed immaginabili diventano monitorabili.

Nagios, inoltre, si basa sull'utilizzo di altri due moduli, per la precisione NRPE e NSCA, che sono, più precisamente, due client di nagios.
NRPE è l'acronimo di Nagios Remote Plugin Executor. In pratica viene utilizzato su un computer remoto per il controllo delle risorse della macchina. Alcuni "plugin", come "check_disk" o "check_load", sono in grado di monitorare servizi solo sulla macchina sui cui girano.
NRPE esegue, sul computer su cui gira, il "plugin" richiesto da Nagios e invia al sistema di monitoraggio il risultato della sua esecuzione.
Il risultato viene mandato attraverso la rete, volendo criptando i dati trasmessi, e viene accettato dal sistema di monitoraggio. NRPE lavora sulla porta TCP 5666.
In linea di massima NRPE va ha controllare le risorse del sistema in modo da poter predire eventuali problemi o crash del sistema.

NSCA è l'acronimo di Nagios Service Check Acceptor. Questo programma serve per creare un sistema di monitoraggio distribuito.
Il pacchetto di questo "agent" di Nagios è formato da un "deamon" e un client. Viene installato un Nagios sul server distribuito e qui viene posto il "client" di NSCA (send_nsca) che spedisce al Nagios centrale tutti i risultati ottenuti. Sul server centrale di Nagios viene installato il "daemon" di NSCA che rimane "in ascolto" sulla porta TCP 5667. Quando arrivano i dati, il "daemon" li accetta e li propone al sistema di monitoraggio come "passive checks", cioè

come controlli che il processo di Nagios non produce attivamente ma di cui riceve solo il risultato ottenuto.

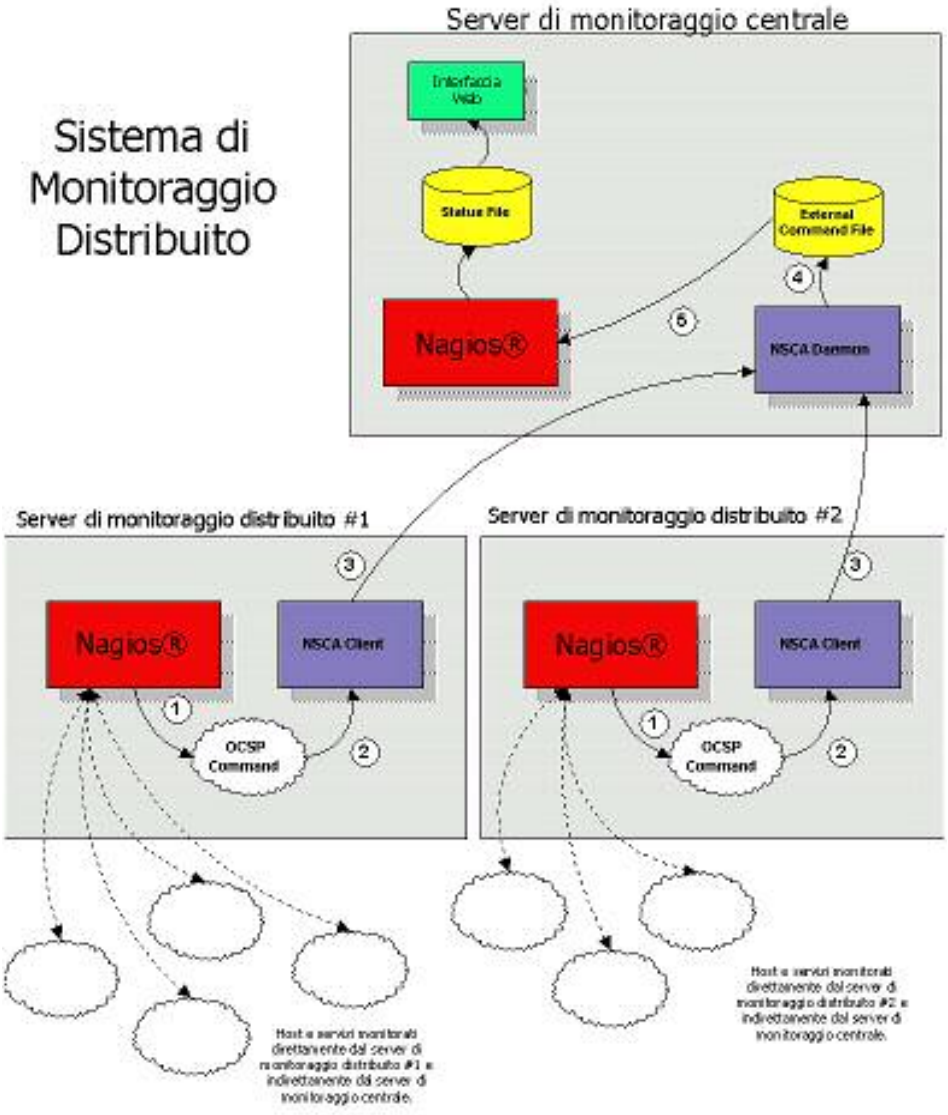

Fig. 6 – Schema di un sistema di monitoraggio distribuito

Un sistema di server di monitoraggio distribuito serve nel caso in cui non sia possibile arrivare ad alcuni computer perché, ad esempio, posti dietro un "firewall".

Impostare tutte le regole necessarie per arrivare a tutti i terminali che si vogliono monitorare con dei controlli attivi da parte del

Nagios centrale, oltre a essere alquanto scomodo, potrebbe risultare controproducente per la sicurezza.
Infatti aprire una porta per ogni servizio che si vuole monitorare e questo per ogni macchina vuol dire aprire tante porte e di conseguenza altrettante vie per sfruttare le possibili vulnerabilità.
Con un sistema di monitoraggio distribuito, invece, possono essere controllati tutti i servizi e tutte le macchine poste dietro un firewall senza dover aprire un numero spropositato di porte, ma solo un canale, tra l'altro criptata tra due server Nagios.

2.2 I protocolli utilizzati

Innanzitutto specifichiamo cos'è un protocollo. Un protocollo è una serie di regole su come comporre dei messaggi che devono essere scambiati fra due computer o fra due macchine in generale. Può essere molto dettagliato, specificando il significato di ogni singolo bit trasmesso, oppure può specificare come far trasferire interi file. In quest'ultimo caso è importante sottolineare come un protocollo che specifica come inviare interi file si appoggi sulle regole specificate in un altro protocollo che definisce come inviare i singoli bit.
Sicuramente il protocollo più importante utilizzato per il monitoraggio di una rete lan è quello che specifica come devono essere inviati i pacchetti all'interno di una rete: il TCP/IP.
Ma è di una certa rilevanza anche un altro protocollo: SNMP.

2.2.1 Il protocollo TCP/IP

Il TCP/IP consiste in una famiglia di protocolli strettamente dipendenti l'uno dall'altro, fra i quali TCP e IP che sono gli acronimi

di Transmission Control Protocol e Internet Protocol. Inizialmente sviluppato per mainframe, il protocollo TCP/IP fu portato sui sistemi Unix alla fine degli anni '70, diventando parte integrale di quel sistema operativo. Oggi, con il diffondersi di Internet, è disponibile su tutti i personal computer.

Ogni macchina all'interno di una rete è individuata univocamente da un indirizzo formato da 32 bit suddivisibili in 4 byte. Perciò possiamo dividere i 32 bit in quattro numeri decimali che vanno da 0 a 255.

Vengono poi individuate tre classi di indirizzi denominate A, B e C.

```
 xxxxxxxx     xxxxxxxx xxxxxxxx xxxxxxxx
<--rete-->  <-----------computer----------->
```

La classe A presenta da 0 a 127 come primo numero che specifica il numero di rete mentre gli altri tre byte identificano il computer. Questa classe dà la possibilità di identificare poche reti ma per ognuna di queste si possono creare 2^{24}-2=16.777.214 nodi: al numero intero vengono sottratti due nodi perché il primo numero disponibile rappresenta il computer e l'ultimo rappresenta la rete.

Ad alcune aziende è stato assegnato un intero blocco di indirizzi di classe A e, di conseguenza, anche i più di sedici milioni di nodi corrispondenti: ad esempio la HP ha gli indirizzi 15.0.0.0 mentre la Apple ha la 17.0.0.0.

```
 xxxxxxxx xxxxxxxx    xxxxxxxx xxxxxxxx
<--------rete--------> <-----computer----->
```

La classe B invece presenta da 178 a 191 come primo byte. Inoltre i primi due byte identificano la rete mentre i secondi due identificano il computer; possono quindi essere identificate 2^{14}-2=16.382 reti e per ognuna di queste possono essere identificati 2^{16}-2=65.534 nodi.

xxxxxxxx xxxxxxxx xxxxxxxx xxxxxxxx
<----------rete----------> <-computer->

La classe C infine presenta da 192 a 223 come primo numero. Poiché i primi tre bit sono fissi, ci sono 2^{21}-2=2,097,150 possibili reti ognuna delle quali ha la possibilità di identificare 256-2=254 macchine.
Infine sono stati riservati tre gruppi di indirizzi per le reti private

- da 10.0.0.0 a 10.255.255.255 (una singola rete di classe A)
- da 172.16.0.0 a 172.31.255.255 (16 reti contigue di classe B)
- da 192.168.0.0 a 192.168.255.255 (256 reti contigue di classe C)

Questi indirizzi vengono utilizzati per configurare reti interne non accessibili dall'esterno.

Ma non basta identificare in modo univoco una macchina con un indirizzo ip. È normale che su una macchina siano presenti più applicativi contemporaneamente. Quando un messaggio viene mandato ad un indirizzo come può la macchina capire automaticamente a quale applicativo direzionarlo? Per ovviare a questo problema è stato introdotto il concetto di "porte"come un punto di accesso a un'applicazione nel sistema. Si tratta in pratica di un'estensione del concetto di porta hardware. Una porta tcp è identificata da un numero.
Le porte dalla 1 alla 1023 sono state assegnate a servizi standard come ad esempio il pop sulla 110, smtp sulla 25, dalla 1024 alla 32767 sono disponibili per servizi non standard ma proprietari come

MySQL o i server Domino ed infine le porte sopra la 32767 sono utilizzate dinamicamente per identificare i client come browser internet o client di posta elettronica.

2.2.2 Il protocollo SNMP

SNMP è l'acronimo di Simple Network Management Protocol ed è un protocollo che è stato scritto essenzialmente per i gestori dell'ambiente di rete. Questo protocollo mira a fornire la possibilità di leggere attraverso la rete informazioni di stato e allarmi da periferiche o sistemi collegati in rete.
SNMP prevede l'installazione di un "agent" su ogni elemento da monitorare. È possibile quindi interrogare gli "agent" SNMP per ottenere le informazioni rilevate. Alcune di queste possono essere modificati attraverso la rete dall'amministratore di sistema e possono anche regolare il funzionamento del sistema stesso. Inoltre possono essere gestiti anche messaggi di rilevazione dei guasti che possono essere inviati dagli "agent" quando c'è un malfunzionamento o un qualunque altro evento significativo.
Quindi il compito primario di SNMP è permettere la gestione della rete e scambiare messaggi relativi all'amministrazione.

2.3 Log di Nagios

Nella cartella */usr/local/nagios/var* sono presenti tutti i log suddivisi in differenti file. Il più importante è "nagios.log" dove vengono salvati tutti i log in generale. Ogni giorno a mezzanotte il file viene salvato con la data corrispondente nella sotto-cartella *archives*, in questo modo sarà possibile consultare i vecchi log e controllare il perchè di qualche malfunzionamento passato. In questi file sono

presenti tutte le notifiche che Nagios manda e i problemi che riscontro nei controlli sui vari servizi.
In questi log sono presenti anche tutti gli "external command", in altre parole i risultati che un nagios distribuito manda al server di monitoraggio centrale.
Infine sono presenti i log generati da un malfunzionamento del sistema di monitoraggio. Quando Nagios non funziona l'unico modo per trovarne la causa è cercare fra questi log.
Sono presenti inoltre altri tre file "comment.log", "downtime.log" e "status.sav". Nel primo troviamo i vari commenti generati automaticamente da Nagios o scritti da un utente per spiegare il malfunzionamento di un host o di un servizio.
In "downtime.log" sono presenti i periodi di momentanea sospensione di monitoraggio di un servizio o di un host con il corrispettivo commento di spiegazioni. Queste sospensioni avvengono generalmente per impedire a Nagios di inviare notifiche quando è stato preventivato un intervento di manutenzione di un certo host che per quel periodo sarà irraggiungibile.
L'ultimo file, "status.sav", è utilizzato da Nagios per salvare lo stato degli host prima di terminare il processo in modo tale che al riavvio possa essere utilizzato dall'interfaccia web per non far vedere che i servizi sono in attesa di essere controllati.
Quando viene compilato Nagios può essere attivata la versione con l'interfacciamento ad una banca dati, come può essere MySQL. In questa versione questi file non esistono e i log vengono salvati nel database in apposite tabelle.
Infine c'è anche "nagios.lock". Nagios, soprattutto quando ha molti host e servizi da controllare, tende a creare processi figlio. In questo file è salvato il numero identificativo associato dal sistema

operativo al processo padre Nagios. Quando si decide di arrestare il sistema di monitoraggio si utilizza "nagios.lock" per fermare il processo padre e di conseguenza anche tutti i suoi processi figlio.

2.4 Notifiche via e-mail e via sms

Un sistema di monitoraggio perde la sua funzione primaria se non ha la possibilità di avvisare chi è adibito all'amministrazione della rete. La notifica classica di un sistema di monitoraggio è la e-mail ma, dopo l'esplosione della telefonia mobile, è diventato necessario poter comunicare via sms eventuali problemi essenzialmente per un problema di reperibilità: con il cellulare si è sempre raggiungibili.
Questi sono i due metodi di notifica più utili e usati. Ne esiste un terzo, in realtà, meno usato ma ugualmente interessante: la notifica via Istant messenger come ICQ, Yahoo Messenger o Jabber.

2.4.1 Notifiche via e-mail

Le notifiche via e-mail costituiscono il metodo classico con cui un sistema di monitoraggio comunica un problema all'amministratore di sistema.
Nagios può essere configurato per mandare e-mail di notifica per problemi sugli host e sui servizi.

Per quanto riguarda gli host si può configurare un comando "host-notify-by-email"

```
define command{
        command name    host-notify-by-email
        command line      /usr/bin/printf "%b" "***** Nagios   *****Type:
$NOTIFICATIONTYPE$: $HOSTNAME$: $HOSTSTATE$: $HOSTADDRESS$: $OUTPUT$/Time:
$DATETIME$" | /bin/mail -s "Host $HOSTSTATE$ alert for $HOSTNAME$!"
$CONTACTEMAIL$
}
```

Questo comando manda una e-mail ai contatti adibiti ad amministratori di rete spiegando quale host ha un determinato stato, in che data e eventuali altre informazioni che Nagios produce automaticamente.
Quindi si può configurare il sistema in modo tale da mandare una e-mail quando il particolare host è in uno dei seguenti stati:

- -d Host in stato "down" (non risponde)
- -u Host in stato "unreachable" (non può essere raggiunto)
- -r Host in "recovery" (la situazione ritorna alla normalità dopo che l'host è stato o "down o "unreachable")

Per quanto riguarda i servizi da monitorare viene configurato il comando "notify-by-email"

```
# 'notify-by-email' command definition
define command{
        command_name    notify-by-email
        command_line      /usr/bin/printf "%b" "***** Nagios   *****Type:
$NOTIFICATIONTYPE$:     $SERVICEDESC$:     $HOSTALIAS$:     $HOSTADDRESS$:
$SERVICESTATE$/Time:   $DATETIME$Info:$OUTPUT$"   |   /bin/mail   -s   "**
$NOTIFICATIONTYPE$ alert - $HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **"
$CONTACTEMAIL$
}
```

Questo comando manda una e-mail spiegando su quale servizio è presente un problema, il numero e l'indirizzo dell'host in questione, la data e eventuali altre informazioni che Nagios crea automaticamente.
Inoltre si può configurare Nagios per avvertire con una e-mail quando il servizio è in uno dei seguenti stati:

- -w Servizio in stato “warning” (i risultati ottenuti dai “plugin” hanno riscontrato un livello di attenzione medio)
- -u Servizio in stato “unreachable” (il servizio non è raggiungibile)
- -c Servizio in stato “critical” (i risultati ottenuti dai “plugin” hanno riscontrato un livello di attenzione alto oppure il servizio non ha risposto alle richieste fatte)
- -r Servizio in “recovery” (la situazione è tornata alla normalità dopo che il servizio è stato “down” o “unreachable”)

2.4.2 Notifiche via sms: smslink

Per inviare una notifica di un problema via sms c'è bisogno di un hardware aggiuntivo: un modem gsm. Il modem utilizzato per l’implementazione del progetto in azienda è un A2D prodotto dalla Falcom. A tutti gli effetti similare ad un cellulare GSM Dual Band, caratteristica comune a tutti i telefonini in commercio, e con l'apposita antenna è in grado di inviare sms ad un qualsiasi numero.

Fig. 7 – Modem Gsm A2D della Falcom

Il modem si connette al computer tramite una semplice porta seriale ed è visto dal personal computer come un comunissimo modem esterno.
Il programma per l'interfacciamento con il modem gsm, anche in questo caso Open Source, si chiama "smslink" e dà la possibilità di inviare sms.
Il programma implementa due moduli un client ed un server. Il server rimane in ascolto su una porta TCP specificata e crea un "processo figlio" per ogni connessione in arrivo. Questi "processi figlio" sono adibiti a raccogliere tutti i parametri per poi accedere al modem gsm.
Il dialogo con il modem gsm è prodotto dal client, rimane connesso fino alla fine del processo di invio del messaggio e fino a quando non riceve un messaggio che certifichi l'invio del sms con successo.
Quindi per mandare un sms si usa il client del programma che è "sendsms".
Si definisce nell'apposito file di configurazione un nuovo comando che fornisce la possibilità di mandare notifiche via sms per problemi riguardanti host o servizi. Definire questo comando è equivalente a scrivere su una riga di comando di una shell di Linux.

```
# 'notify-by-sms' command definition
define command{
        command_name    notify-by-sms
        command_line            /usr/src/smslink-0.55b/client/sendsms   -d
"$CONTACTPAGER$"  -m  "Nagios:  $NOTIFICATIONTYPE$  Service:  $SERVICEDESC$
Host:  $HOSTALIAS$  Indirizzo:  $HOSTADDRESS$  Stato:  $SERVICESTATE$  Data:
$DATETIME$ Info:$OUTPUT$" 127.0.0.1
}
```

Quindi ci sono differenti dati che vengono passati al client "sendsms". Innanzitutto dopo il -d il numero a cui mandare gli sms. Dopo il -m troviamo il messaggio. Ricordandosi che un sms è formato al massimo da 160 caratteri si può mettere il tipo di notifica

il servizio o l'host che presenta problemi, l'indirizzo dell'host, la data e l'eventuale output generato da Nagios. Infine bisogna aggiungere l'indirizzo del server di "smslink" a cui "sendsms" manderà il messaggio generato.
Inviare sms quindi è il modo migliore per far comunicare all'amministratore della rete la presenza di un problema critico. Non è conveniente usare un sms in ogni caso. Con le e-mail si possono mandare tutte le notifiche anche quelle meno importanti, anche perché il volume di posta generato anche se notevole, è facilmente gestibile con un filtro per lo smistamento in apposite cartelle.
Gli sms invece devono essere usati solo per problemi critici, che magari sono in questo stato da molto tempo oppure quando il sistema di email non è funzionante. Questo può essere programmato tramite l'apposito file di configurazione che da la possibilità di differenziare le notifiche.

2.4.3 Integrazione fra notifiche via e-mail e via sms

Mandare, ogni volta che viene inviata una notifica, sia un sms che una e-mail potrebbe essere molto controproducente. Come già è stato detto mentre per le e-mail basta un semplice filtro per smistare la posta in apposite cartelle, lo stesso numero di sms sarebbe difficilmente gestibile da un semplice cellulare.
Per ovviare a questo problema si può utilizzare il file di configurazione "escalation.cfg" dove è possibile impostare Nagios in modo che fino ad una certo numero di notifiche vengano mandate e-mail e dopo anche solo un paio di sms.
Purtroppo Nagios predispone la modalità di invio della notifica ("notify-by-email" o "notify-by-sms") nel file di configurazione

contats.cfg. Questo significa che saranno mandati lo stesso numero di sms ed e-mail, a meno che non si utilizzi uno stratagemma.
Per ogni persona fisica a cui mandare le notifiche verranno creati due contatti nel file contats,cfg: uno per le notifiche via e-mail e l'altro per gli sms. Dopodiché nel file di configurazione "contatgroups.cfg" saranno creati due gruppi di contatti anche qui uno per le e-mail e l'altro per gli sms.

```
# 'nagios' contact definition
define contact{
        contact_name                    guido-email
        alias                           Guido
        service_notification_period     24x7
        host notification period        24x7
        service notification options    w,u,c,r
        host_notification_options       d,u,r
        service_notification_commands   notify-by-email
        host_notification_commands      host-notify-by-email
        email                           indirizzo@email.it
        pager                           3491231231
        }
```

```
# 'nagios' contact definition
define contact{
        contact_name                    guido-sms
        alias                           Guido
        service notification period     24x7
        host notification period        24x7
        service_notification_options    w,u,c,r
        host_notification_options       d,u,r
        service notification commands   notify-by-sms
        host notification commands      host-notify-by-sms
        email                           indirizzo@email.it
        pager                           3491231231
        }
```

Infine nelle configurazioni del file escalation.cfg basta immettere uno dei due differenti gruppi quando si vuole che lo specificato problema sia notificato in un modo o nell'altro.

```
# Serviceescalation definition
define serviceescalation{
        host_name                       fw01,fw02
        service_description             SSH
        first notification              2
        last notification               5
        contact groups                  email
        notification interval           30
        }
```

```
# Serviceescalation definition
define serviceescalation{
        host name                       fw01,fw02
        service_description             SSH
        first_notification              7
        last notification               8
        contact groups                  sms
        notification interval           120
        }
```

Capitolo 3

CONFIGURAZIONI

3.1 Nagios

Tutti i file di configurazione di Nagios sono sotto la cartella */usr/local/nagios/etc*.
Il più importante, vero e proprio cuore del programma, è nagios.cfg. Contiene, infatti, le più importanti direttive che Nagios deve seguire e viene letto sia dal programma vero e proprio che dai file CGI, che creano l'interfaccia HTML dinamica.
Fra le caratteristiche più importanti di questo file vi è la possibilità di attivare o disattivare le notifiche, impostare la posizione degli altri file di configurazione o come e quali log salvare.
Inoltre abbiamo il file cgi.cfg in cui troviamo le impostazioni riguardanti l'interfaccia HTML. Sono presenti le impostazioni per interfacciarsi ad un eventuale database MySQL, il modo in cui l'interfaccia deve generare i grafici rappresentanti l'andamento dei servizi o degli host e le mappe per la descrizione delle reti monitorate.
Infine troviamo anche i permessi per i vari utenti che hanno la possibilità di controllare i risultati ottenuti da Nagios tramite l'interfaccia HTML.
Ma per Nagios le configurazioni più importanti sono quelle riguardanti gli host e i servizi. Questi file sono basati su "template", ovvero modelli sui cui si basano tutte le configurazioni degli "Object Data", dando la possibilità di creare differenti tipi di host o servizi da monitorare.

Per gli host le configurazioni vengono salvate nel file "hosts.cfg":

```
# Generic host definition template

define host{

        name                        generic-host  ; The name of this host

        event_handler_enabled           1       ; Host event handler
        flap detection enabled          1       ; Flap detection
        process perf data               1       ; Process performance data
        retain status information       1       ; Retain status
        retain_nonstatus_information    1       ; Retain non-status

        register                        0       ; DONT REGISTER THIS



        }
```

Questo è il template presente nel file di default di host.cfg e dà la possibilità di configurare gli host con il nome "generic-host". Questo significa che gli host configurati con quel nome avranno tutte le caratteristiche specificate nel ”template”. In questo caso ad esempio avranno attivate le notifiche, il “flap detection” che disabilita le notifiche in caso di un continuo passaggio di stato da up a down e il retain_status che salva lo stato degli host per quando si chiude Nagios.

Importante è non impostare ad 1 la voce "register" perché, in questo caso, non sarebbe più un template ma diventerebbe un host.

```
# 'assisteza2' host definition

define host{
        use                     generic-host; Name of host template to use



        host_name               assistenza2
        alias                   Assistenza 2 BLS
        address                 172.0.1.55
        parents                 hub2
        check_command           check-host-alive
        max_check_attempts      10
        notification_interval   120
        notification period     24x7
        notification options    d,u,r

}
```

```
# 'fw01' host definition
define host{
        use                     generic-host; Name of host template to use

        host name               fw01-bls
        alias                   Firewall 01 BLS
        address                 172.30.0.1
        parents                 assistenza.bls.it
        check_command           check_ssh
        max check attempts      10
        notification interval   120
        notification period     24x7
        notification options    d,u,r
        }
```

Questi invece sono esempi di configurazioni di host. Sono entrambi "generic-host", un template che in linea di massima è adatto alla maggior parte dei casi. In più per ogni host viene assegnato un nome, host_name, con il quale sarà identificato all'interno di tutte le configurazioni di Nagios, un alias che spiega, in poche parole, le caratteristiche dell'host e un indirizzo.

La riga "check_command" è molto importante; qui infatti troviamo il nome del comando che Nagios utilizza per determinare lo stato dell'host: se il risultato del controllo è positivo l'host sarà up. Nel primo caso troviamo "check-host-alive" che è un comando definito nel file "checkcommands.cfg", che manda un ping all'indirizzo dell'host.

Nel secondo caso, invece, l'host in questione è un firewall e quindi, per motivi di sicurezza, non risponde ai ping. Di conseguenza il comando check-host-alive, che è basato sui ping, è inutile. Per ovviare a questo problema, basta definire nel file "checkcommands.cfg" un nuovo comando, in questo caso "check_ssh", basato su un servizio che è noto essere attivo e raggiungibile da internet sul "firewall".

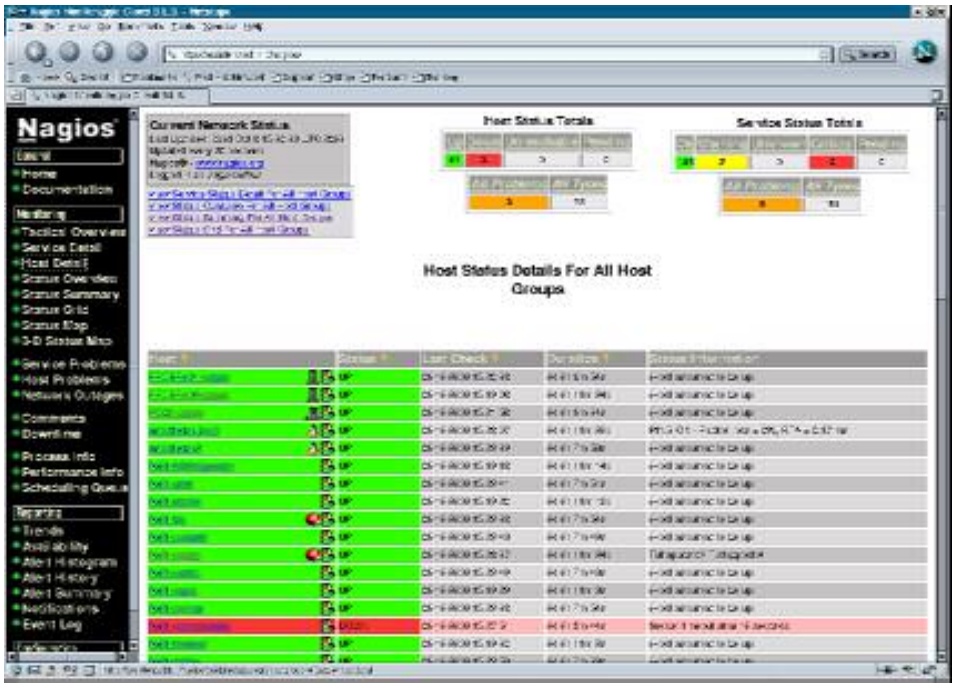

Fig. 8 – Elenco degli host monitorati

Nagios farà dieci tentativi prima di notificare il problema. Le notifiche saranno mandate ogni 120 minuti, ovvero ogni due ore.
Inoltre gli host sono raggruppati in quello che Nagios definisce hostgroups.

L'idea di utilizzo è similare per quel che riguarda il file "services.cfg". Anche qui abbiamo la possibilità di definire del "template": in questo caso ne abbiamo due.

```
# Generic service definition template
define service{
        name                            generic-service
        active_checks_enabled           1       ; Active service checks
        passive_checks_enabled          1       ; Passive service checks
        parallelize_check               1       ; Active service checks
        obsess_over_service             1       ; We should obsess over
        check_freshness                 1       ; Default is to NOT check
        notifications_enabled           1       ; Service notifications
        event_handler_enabled           1       ; Service event handler
        flap_detection_enabled          1       ; Flap detection is enable
        process_perf_data               1       ; Process performance data
        retain_status_information       1       ; Retain status
        retain_nonstatus_information    1
        register                        0
        }
```

```
# Passive service definition template
define service{
        name                            passive-service  ; Template name
        active checks enabled           0       ; Disable Active checks
        passive_checks_enabled          1       ; Enable Passive checks
        parallelize_check               1       ; parallelize checks
        obsess_over_service             1       ; obsess over this svc
        check freshness                 1       ; check service fresh
        freshness threshold             1800    ; Stale if over 30 min.
        notifications enabled           1       ; enable notification
        event_handler_enabled           1       ; enable event handler
        flap_detection_enabled          1       ; enable flap detection
        process_perf_data               1       ; Process performance data
        retain status information       1       ; Retain status info
        retain nonstatus information    1       ; Retain non-status info
        check command   no-passive-update       ; if stale run this cmd
        register                        0       ; DONT REGISTER THIS
        }
```

Anche in questo caso come nel caso degli host questi template definiscono una configurazione comune a tutti i servizi con quel nome. In questo caso i due tamplate differiscono solo per un termine che è active_check_enabled. Infatti il primo è un template per i servizi che vengono controllati direttamente dal processo centrale di Nagios, è in realtà il template di default sul quale si basano tutti i servizi. Il secondo invece è il template per i servizi passivi. In pratica tutti i servizi che non sono controllati direttamente dal Nagios "centrale" ma dai Nagios distribuiti. Quindi vengono mandati i risultati via internet e il daemon di NSCA li propone a Nagios come external check, ovvero passive check.

```
# Service definition
define service{
        use                              generic-service         ; Name of
service template to use


        host name                       assistenza.bls.it
        service_description             PING
        is_volatile                     0
        check_period                    24x7
        max check attempts              3
        normal check interval           5
        retry check interval            1
        contact_groups                  nagiosgroup
        notification_interval           120
        notification period             24x7
        notification options            c,r
        check command                   check ping!700.0,20%!800.0,60%
}
```

Questo è invece un esempio di configurazione di un servizio. Il servizio è il ping controllato sull'host assistenza.bls.it, che ha un indirizzo univoco nel file hosts.cfg. Nagios farà il controllo del servizio per tre volte, ad intervalli di un minuto nel caso non abbia un riscontro positivo immediato e in caso negativo manderà una notifica. Le notifiche saranno mandate a distanza di 120 minuti, ovvero ogni due ore, l'una dall'altra.

Nel caso in cui tutto è andato bene Nagios è configurato per controllare il servizio ogni cinque minuti.

Infine c'è la riga "check_command" che è il comando a cui è associato il servizio da controllare.

In questo caso abbiamo: `check_ping!700.0,20%!800.0,60%`

Check_ping è il nome del plugin che Nagios lancerà su riga di comando e i dati che sono presenti dopo il punto esclamativo sono i parametri per stabilire in che stato è il servizio: ok, warning o critical.

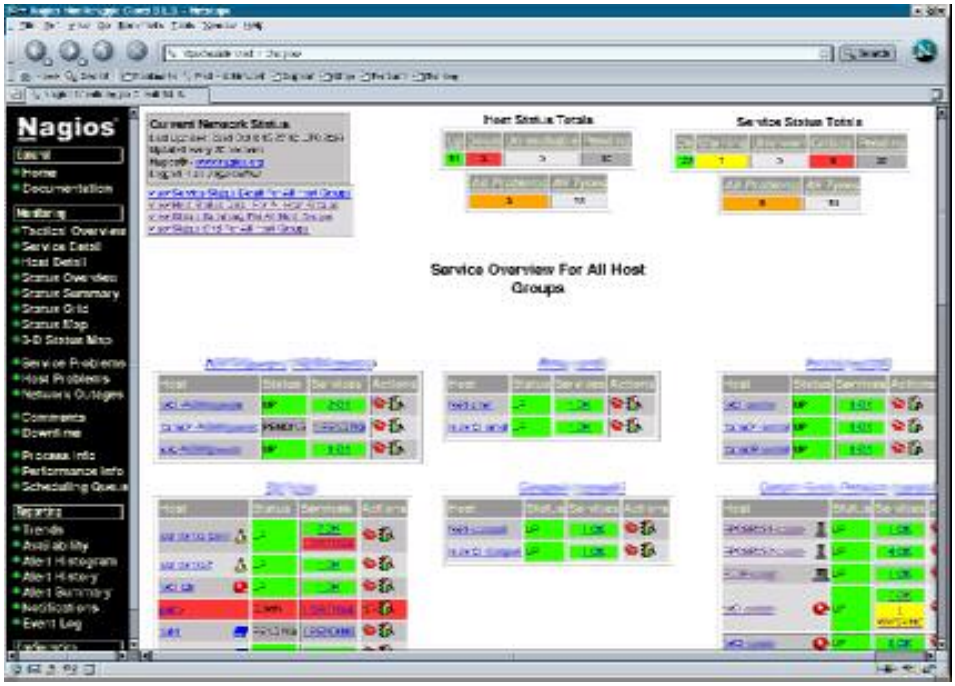

Fig. 9 – Elenco di host e servizi monitorati: Status overview

Il comando "check_ping" viene definito nel file "checkcommands.cfg" nel seguente modo:

```
# 'check_# 'check_ping' command definition
define command{
        command name    check ping
        command line     $USER1$/check ping -H $HOSTADDRESS$ -w $ARG1$ -c
$ARG2$ -p 5
        }
```

"Check_ping" ha la seguente sintassi:

```
check_ping -H <host_address> -w <wrta>,<wpl>%% -c <crta>,<cpl>%% [-p
packets] [-t timeout]
```

Il comando comprende l'host a cui mandare i ping, un valore in percentuale di pacchetti persi o di tempo trascorso sia per il livello di warning, sia per il livello di critical.

Infine troviamo altri due file "escalations.cfg" e "dependencies.cfg". Il primo serve per differenziare le notifiche mandate via mail o via sms oltre a fare in modo di mandarle fino ad un certo numero.
Il secondo invece imposta le dipendenze fra i vari host in modo tale che se per caso il firewall diventa, per qualsiasi ragione, irraggiungibile, Nagios non manderà notifiche per tutti gli host della rete ma solo per quel problema.

3.2 NRPE

NRPE è l'acronimo di Nagios Remote Plugin Executor ed è il client che esegue in remoto i "plugin" e riporta i risultati al sistema di monitoraggio.

NRPE è formato da un programma che lavora sulla porta 5666 e dall'eseguibile "check_nrpe" presente sul server di management che creerà le richieste opportune da mandare all'host remoto.
NRPE ha in realtà un solo file di configurazione, "nrpe.cfg". In questo file possiamo impostare la porta su cui NRPE deve ricevere i dati e quali indirizzi hanno la possibilità di richiedere i risultati dei controlli in modo tale da controllare chi richiede informazioni sul terminale in questione.
Ma la parte più importante è la definizione dei comandi.

```
command[check_disk1]=/usr/local/nagios/libexec/check_disk -w 20 -c 10 -p
/dev/hda1
```

Ogni comando viene prima definito in questo file e poi richiesto dal Nagios centrale. Se il comando non è stato definito in questa sezione NRPE non eseguirà il controllo richiesto e non fornirà alcun dato.
In questo caso è stato definito il comando check_disk1. Questo check controlla la quantità di spazio libero sulla partizione hda1 che, nella maggioranza dei sistemi Linux, è definita sul disco fisso "master", su cui, in pratica, viene caricato parte del sistema operativo. Come tutti i comandi di Nagios è presente un valore di "warning" e un valore di "critical". Questa è la configurazione base per poter eseguire i "plugin" in remoto.
Naturalmente è necessario dover configurare anche Nagios per poter inviare i comandi giusti al processo NRPE presente sull'host remoto.
Innanzitutto è necessario aggiungere una definizione di comando nel file di configurazione "checkcommands.cfg":

```
# Definizione per nrpe
define command{
command name check nrpe
command_line   /usr/local/nagios/libexec/check_nrpe   -H   $HOSTADDRESS$   -c
$ARG1$
}
```

Quindi Nagios creerà un comando che comprende l'indirizzo a cui chiedere il controllo ed il comando che si sarà definito nel file di configurazione di NRPE.
Un esempio pratico potrebbe essere:

```
/usr/local/nagios/libexec/check_nrpe -H 192.0.0.10 -c check_disk1
```

Quindi NRPE, presente sull'host corrispondente all'indirizzo 192.30.2.10, eseguirà il comando check_disk1 e riporterà quanto spazio libero è presente nella partizione hda1.
Infine è importantissimo configurare nel modo adeguato la definizione di servizio presente nel file services.cfg.
Un esempio pratico potrebbe essere:

```
# Service definition
define service{
        use                             generic-service         ; Name of
service template to use

        host_name                       assistenza.bls.it
        service description             DISCO
        is volatile                     0
        check period                    24x7
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            1
        contact groups                  nagiosgroup
        notification interval           120
        notification period             24x7
        notification_options            c,r
        check_command                   check_nrpe!check_disk1
}
```

La configurazione di questo servizio è in tutto e per tutto equivalente ai servizi normali. Cambia però la linea corrispondente a check_command. Qui è presente il comando

```
check nrpe!check disk1
```

Che equivale ad eseguire il comando check_nrpe con check_disk1 come argomento all'opzione –c.

3.3 NSCA

NSCA è l'acronimo di Nagios Service Checks Acceptor e viene utilizzato per poter configurare una rete di Nagios distribuiti da porre dietro ai firewall. Il modulo è formato dal programma NSCA che rimane in ascolto sulla porta 5667 e dall'eseguibile "send_nsca" presente sul Nagios distribuito. Entrambi questi file hanno il loro file di configurazione, rispettivamente "nsca.cfg" e "send_nsca.cfg".
In "nsca.cfg" troviamo le configurazioni più importanti da impostare tra le quali la porta TCP dove il processo si metterà in ascolto, quali sono gli indirizzi degli host distribuiti da cui accettare i risultati dei "check" e le configurazioni di crittografia.
In particolare nel file di configurazione "send_nsca" sono presenti impostazioni solo riguardanti la crittografia. Infatti è possibile scegliere il metodo di crittografia e la password da associare. In questo modo si rende possibile la lettura dei dati che corrono lungo la rete solo ad NSCA impedendone l'intercettazione.
Ma le impostazioni presenti in questo due file non sono tuttavia sufficienti. È necessario impostare adeguatamente anche entrambi i Nagios, quello distribuito e quello centrale.
Per il Nagios distribuito è necessario impostare nel file "nagios.cfg" la riga "enable_notification" a 0 in modo da impedire che vengano inviate notifiche. Inoltre, bisogna cambiare la configurazione di obsess_over_services da 0 a 1 in modo tale da continuare a

controllare i servizi in modo "ossessivo" e inviare i dati tramite l'ocsp_command.

Quindi bisogna, sempre nel file nagios.cfg, impostare il comando ocsp_command=submit_check_result in modo che ogni volta che Nagios completa un controllo su un servizio mandi il risultato al Nagios centrale.

Inoltre bisogna creare, sotto la cartella */usr/local/nagios/libexec/eventhandlers*, lo script eseguibile submit_check_result in questo modo:

```
#!/bin/sh
# Arguments:
# $1 = host name (Short name of host that the service is
# associated with)
# $2 = svc_description (Description of the service)
# $3 = state_string (A string representing the status of
# the given service - "OK", "WARNING", "CRITICAL"
# or "UNKNOWN")
# $4 = plugin output (A text string that should be used
# as the plugin output for the service checks)
#
# Convert the state string to the corresponding return code
return_code=-1
case "$3" in
        OK) return_code=0 ;;
        WARNING) return_code=1 ;;
        CRITICAL) return_code=2 ;;
        UNKNOWN) return_code=-1 ;;
esac
# pipe the service check info into the send_nsca program, which
# in turn transmits the data to the nsca daemon on the central
# monitoring server

/bin/echo -e "$1\t$2\t$return_code\t$4\n" |
/usr/local/nagios/bin/send_nsca central_server -c
/usr/local/nagios/etc/send_nsca.cfg
```

Importante è l'ultima riga: tramite l'eseguibile send_nsca manda i risultati del controllo effettuato al "central server" utilizzando il file di configurazione send_nsca.cfg.

Infine bisogna aggiungere nel file checkcommands.cfg la definizione del comando come segue:

```
define command{
command_name submit_check_result
command_lin     /usr/local/nagios/libexec/eventhandlers/submit_check_result
$HOSTNAME$ '$SERVICEDESC$' $SERVICESTATE$ '$OUTPUT$'
}
```

che passa come argomenti all'eseguibile submit_check_result il nome dell'host, il nome del servizio, lo stato del servizio e l'output che Nagios crea automaticamente.
Per quanto riguarda il Nagios centrale è necessario:

- Impostare "enable_notification" ad 1
- Attivare "external_command_check"
- Attivare i "passive_ check"

In più bisogna definire tutti i servizi e gli host che vengono controllati dal Nagios distribuito e quindi non attivamente.
I servizi verranno impostati in modo da non essere attivamente eseguiti e saranno definiti come "passive_check".

3.4 SNMP

Nagios può generare degli alert dalla ricezione dei trap inviati da un agent SNMP.
Per spiegare al meglio l'interazione fra Nagios e un agent SNMP è necessario utilizzare un esempio pratico. In questo caso si è presa la situazione del monitoraggio di un servizio denominato "ArcServe Backup" su un server Novell. Un servizio che genera degli alert per SNMP è impostato come segue:

```
define service{
        host name                       novellserver
        service description             ArcServe Backup
        is_volatile                     1
        active_checks_enabled           0
        passive_checks_enabled          1
        max check attempts              1
        contact groups                  novell-backup-admins
        notification interval           120
        notification_period             24x7
        notification_options            w,u,c,r
        check_command                   check_none
}
```

Importante è sottolineare l'opzione "is_volatile" impostata ad 1 in modo tale che per tutti gli alert che arrivano venga creata una notifica. Inoltre sono disabilitati gli "active_check" mentre i "passive_check" sono attivati in modo che i risultati dei controlli arriveranno dal NSCA presente sulla macchina dove è presente l'agente SNMP.
Sulla macchina che viene monitorata da SNMP è necessario

- Modificare l'autopilot per mandare i "trap" di SNMP
- configurare SYS:\ETC\TRAPTARG.CFG e aggiungere l'indirizzo della macchina che riceverà i trap di SNMP
- Caricare SNMP.NLM
- Caricare ALERT.NLM per facilitare l'invio dei trap in tempo reale

Sulla macchina di Management di SNMP si installa il software Net-snmp quindi:

- Installare ArcServe MIBs
- Impostare il file di configurazione di snmptrapd per utilizzare gli alert di ArcServe
- Far partire snmptrapd per accettare trap di SNMP in arrivo.

In modo che snmptrapd inoltri i trap di SNMP al processo di Nagios dobbiamo impostare il file */etc/snmp/snmptrapd.conf*
In questo modo:

```
##############################
# ArcServe SNMP Traps
##############################
```

```
# Tape format failure
straphandle ARCserve-Alarm-MIB::arcServetrap9
/usr/local/nagios/libexec/eventhandlers/handle-arcserve-trap 9

# Failure to read tape header
traphandle ARCserve-Alarm-MIB::arcServetrap10
/usr/local/nagios/libexec/eventhandlers/handle-arcserve-trap 10
# Failure to position tape

traphandle ARCserve-Alarm-MIB::arcServetrap11
/usr/local/nagios/libexec/eventhandlers/handle-arcserve-trap 11

# Cancelled jobs
traphandle ARCserve-Alarm-MIB::arcServetrap12
/usr/local/nagios/libexec/eventhandlers/handle-arcserve-trap 12

# Successful jobs
traphandle ARCserve-Alarm-MIB::arcServetrap13
/usr/local/nagios/libexec/eventhandlers/handle-arcserve-trap 13

# Imcomplete jobs

traphandle ARCserve-Alarm-MIB::arcServetrap14
/usr/local/nagios/libexec/eventhandlers/handle-arcserve-trap 14

# Job failurestraphandle ARCserve-Alarm-MIB::arcServetrap15
/usr/local/nagios/libexec/eventhandlers/handle-arcserve-trap 15
```

Quindi sotto la cartella */usr/local/nagios/libexec/eventhandlers/* si trova l'eseguibile *handle-arcserve-trap* che è uno script eseguibile:

```
#!/bin/sh
# Arguments:
# $1 = trap type
# First line passed from snmptrapd is FQDN of host that sent the trap read
host
# Given a FQDN, get the short name of the host as it is setup in Nagios
hostname="unknown"

case $host in
        novellserver.mylocaldomain.com)
        hostname="novellserver"
        ;;
        nt.mylocaldomain.com)
        hostname="ntserver"
        ;;
        esac

# Get severity level (OK, WARNING, UNKNOWN, or CRITICAL) and plugin output
based on trape type
state=-1
output="No output"
        case "$1" in

        # failed to format tape - critical
        11)
                output="Critical: Failed to format tape"
                state=2
                ;;
        # failed to read tape header - critical
        10)
                output="Critical: Failed to read tape header"
                state=2
                ;;
        # failed to position tape - critical
        11)
                output="Critical: Failed to position tape"
                state=2
```

```
                ;;
        # backup cancelled - warning
        12)
                output="Warning: ArcServe backup operation cancelled"
                state=1
                ;;
        # backup success - ok
        13)
                output="Ok: ArcServe backup operation successful"
                state=0
                ;;
        # backup incomplete - warning
        14)
                output="Warning: ArcServe backup operation incomplete"
                state=1
                ;;
        # backup failure - critical
        15)
                output="Critical: ArcServe backup operation failed"
                state=2
                ;;
esac

# Submit passive check result to monitoring host
/usr/local/nagios/libexec/eventhandlers/submit check result $hostname
"ArcServe Backup" $state "$output"
exit 0
```

Questo script richiama un altro script chiamato submit_check_result che è già stato definito nell'utilizzo di NSCA.

Capitolo 4

SICUREZZA

4.1 I firewall

Un firewall è un computer adibito ad essere l'unico punto di collegamento di due o più reti in modo tale che tutte le comunicazioni fra le due reti dovranno passare per questo terminale.
Il firewall, quindi, è configurato in modo tale da far passare alcuni dati, bloccarne altri e reagire a determinati eventi.
In un firewall Linux, gli strumenti con cui si operano le decisioni sui vari dati sono fornite direttamente dal kernel tramite i comandi "ip" che serve per configurare interfacce di rete, "iptables" utilizzato per bloccare o consentire il traffico ed infine "tc", acronimo di traffic classifier, che ha la funzione di assegnare priorità, imporre precedenze o limiti di banda.
In pratica il firewall difende la rete utilizzando "regole" che sono impostate da chi lo configura; quindi queste regole dovranno tener conto di alcuni accorgimenti.
Innanzitutto il firewall dovrà proteggere la rete in modo tale da rendere accessibili dall'esterno solo alcuni servizi. È di primaria importanza poi che il traffico sui server non ecceda un certo "rate" in modo da non sovraccaricare e quindi interrompere il servizio. Un altro problema potrebbe arrivare dagli utenti della rete nel caso attivino operazioni non consentite: è quindi importante impedire che ciò accada.

Infine, è importante che i server e i terminali all'interno della rete non possano sferrare attacchi verso terzi e limitare i danni che un server "conquistato" da un malintenzionato possa arrecare alla rete. È, infatti, importante prevenire un effetto a catena che provocherebbe la caduta di tutti i server di una rete locale.

Per implementare al meglio queste direttive, un server è solitamente assemblato con tre schede di rete: una destinata alla rete locale, una destinata ad internet e una per quella che è denominata DMZ.

DMZ è l'acronimo di De-Militarized Zone ed è una piccola rete in cui sono messi solamente i server in modo tale da difenderli da eventuali attacchi provenienti dalla rete esterna e dalla rete interna. Allo stesso modo difende la rete interna dalla DMZ e da internet e quest'ultimo dalla rete interna e dalla DMZ.

Come è implementato tutto ciò a livello software? Innanzitutto il firewall lavora tramite il concetto di catene ovvero "ipchains". Il kernel di linux dispone di tre catene di base: quella di input, di forward e di output. Ora per un firewall pensato come sistema di sicurezza per un computer normale o per un server ha senso pensare solamente alle catene di input e output: in pratica viene regolamentata solamente tutto ciò che entra o esce dalla macchina in questione. Per un firewall considerato come sistema di sicurezza per più reti subentra tuttavia il concetto di "forward": inoltrare determinati dati da una rete all'altra.

Ogni catena è poi costituita da un insieme di "regole" ed ad ognuna di questa corrisponde un'azione. Quando arriva un pacchetto sono controllate in ordine tutte le regole, se viene trovato un "match" ovvero se una di queste è applicabile, viene eseguita l'azione

corrispondente e viene terminato il processo. In caso contrario il firewall seguirà le azioni specificate nell'apposita "policy".
La policy è il comportamento che il firewall deve avere quando un pacchetto non trova nessuna regola corrispondente.

- DROP elimina il pacchetto
- ACCEPT accetta il pacchetto
- QUEUE passa il pacchetto ad un altro programma

Di solito il firewall è impostato con il comando "DROP" in quanto una politica che va per esclusione, cioè tutti i pacchetti che arrivano vengono eliminati tranne quelli definiti nelle regole, è molto più sicura rispetto ad una politica che accetta tutti pacchetti e ne vieta solo alcuni.

4.2 Crittografia

Poiché NRPE e NSCA mandano i risultati dei loro controlli attraverso la rete è necessario nascondere questi dati ai numerosi occhi indiscreti che sono presenti in internet. Nonostante questi dati non siano di pari importanza ad un segreto di stato è pericoloso mandare informazioni sulla struttura di una rete locale o sullo stato di salute di un server senza una anche minima protezione. La crittografia dei dati è la risposta più concreta a questo problema.
OpenSSL è un programma Open Source che implementa l'utilizzo de i protocolli SSL e TSL insieme ad un buon numero di librerie apposite utilizzate per la crittografia dei dati.
SSL è l'acronimo di Security Socket Layer ed è un protocollo che fu sviluppato dalla Netscape Developmet Corporation per poi essere accettato universalmente come protocollo per l'autenticazione e la

comunicazione criptata fra client e server. TLS, acronimo di Transport Socket Layer, può essere considerato l'evoluzione di SSL.

SSL si pone i livelli OSI 5 e 6 di Presentation e Session Layer e si pone fra il TCP/IP e i protocolli ad alto livello come HTML o IMAP e consente ad un “client” di autenticarsi ad un server e viceversa, in modo tale da poter stabilire una connessione criptata.

Quindi un SSL server authentication è utilizzato da un utente per autenticare l’identità del server: il client SSL utilizza tecniche standard di crittografia a chiave pubblica per controllare che il certificato di un server sia valido.

Allo stesso modo un SSL authentication client consente ad un server di confermare l'identità di un client: con le stesse tecniche di crittografia a chiave pubblica il server controlla che il certificato prodotto dal SSL client sia valido.

Infine, sarà possibile avviare una connessione SSL criptata dove i file criptati dal software del mittente saranno decriptati dal software ricevente, fornendo un alto grado di confidenzialità: tutto ciò dà la possibilità di evitare che i dati mandati siano manomessi.

SSL, inoltre, include due sotto-protocolli: SSL Record e SSL handshake. SSL record definisce come i dati devono essere inviati mentre SSL handshake utilizza SSL Record per lo scambio di messaggi fra un client e un server la prima volta che è impostata una connessione: viene prima fatta un'autenticazione del server poi vengono paragonati gli algoritmi di cifratura supportati da entrambi, il client si identifica presso il server, viene creata una chiave segreta utilizzata solo per questa connessione ed infine inizia la vera e propria connessione criptata.

SSL utilizza un sistema di chiavi pubbliche e private. Gli algoritmi a chiave simmetrica sono più veloci ma quelli a chiave pubblica che generano una connessione con un'autenticazione più efficace.

SSL Handshake viene fatto grazie a chiavi pubbliche dopodiché il client e il server cooperano per la generazione di una chiave simmetrica che sarà utilizzata per criptare e decriptare i dati che scorrono nel canale SSL.

4.3 Authentication gateway

L'authentication gateway è un servizio che viene utilizzato per poter aprire una regola su un firewall per un determinato periodo di tempo.
In pratica un utente si autentica sul firewall. Da questo momento si apre sul firewall una regola che permette all'ip di accedere ad un determinato servizio.
In pratica questo metodo obbliga l'utente ad autenticarsi e quindi a registrare la propria presenza per accedere ad alcune risorse.
Questo servizio dà la possibilità di accedere a dei servizi da qualsiasi personal computer senza dover toccare le regole poste sul firewall.
Nel caso specifico di Nagios questo si traduce in un mezzo molto importante. Lasciare Nagios disponibile a tutti, coperto solo da una password, potrebbe diventare pericoloso in quanto tutti potrebbero riuscire ad accedervi.
Con l'authentication gateway si implementa un'altra barriera di difesa contro gli occhi indiscreti presenti in internet che potrebbero utilizzare le informazioni di cui dispone Nagios.
In questo caso l'authentication gateway apre solo alle richieste provenienti dall'indirizzo da cui si connette l'utente la porta 80, utilizzata dal webserver Apache per creare le pagine dinamiche di Nagios, in modo tale da rendere possibile il consulto dei dati forniti

dal sistema di monitoraggio da qualsiasi terminale connesso ad internet; questo, senza correre il rischio di lasciare aperto un servizio come quello di Apache Web Server su internet, sì protetto da password, ma a volte anche vulnerabile.

Capitolo 5

CONCLUSIONI

Per quanto riguarda il sistema di monitoraggio implementato in azienda è importante dire che Nagios è sicuramente un'alternativa valida alle varie soluzioni a pagamento di HP o IBM.
Nagios è sicuramente un programma completo e, come ho già ho sottolineato, completamente personalizzabile e di conseguenza completamente adattabile ai repentini cambiamenti che caratterizzano il mondo dell'informatica.
Inoltre in rete è presente un vero e proprio sistema di supporto on-line tramite le mailing-list e i forum che rende la risoluzione dei problemi praticamente immediata.
Certo come tutti i programmi non è esente da bug. Il più evidente si è manifestato quando il programma è stato messo sotto pressione, a monitorare un numero elevato di servizi. In questa situazione il programma tende a non riconoscere immediatamente i cambiamenti impostati nei file di configurazione ma continua ad utilizzare, se pur per un breve periodo, le configurazioni vecchie.
È sicuramente un peccato veniale che, con un po' di pazienza, può essere facilmente superato.
Un pochino più grave è la difficile implementazione del client NRPE per sistemi Windows. Esiste un clone di NRPE denominato NRPE_NT ma ha grossi problemi di implementazione e tende a girare correttamente solo sotto Windows NT mentre per le atre versioni fa molta fatica.

Sul piano personale, i due mesi passati in azienda a lavorare a questo progetto, nonostante il sacrificio delle ferie estive, devo dire essere stati molto utili.

Innanzitutto, cosa a mio avviso basilare per un ingegnere , ho imparato a muovermi con cognizione di causa all'interno del mondo Unix e in particolare Linux.

L'infarinatura per lo più inadeguata che l'università ci propone verso l'informatica pratica mi ha fatto scoprire una soluzione alternativa e soprattutto valida ai sistemi operativi della Microsoft che, mi sento in obbligo di dire, mi ha spinto a cambiare sistema operativo anche al mio personal computer di casa.

L'idea del software libero e privo di copyright a disposizione dell'umanità mi ha sempre affascinato ma devo dire di non aver mai avuto le basi necessarie per compiere il grande salto; ora grazie a ciò che ho imparato in azienda mi sento pronto ad approfondire questo nuovo mondo.

Poi naturalmente sono riuscito ad esplorare il mondo del TCP/IP, anche se non a fondo come vorrei. Certo adesso sono in grado di potermi muovere all'interno di una rete senza creare danni di sorta, ma mi piacerebbe in futuro approfondire maggiormente l'argomento.

Le reti in generale e i firewall mi hanno affascinato più di quanto avessi potuto immaginare alla vigilia del tirocinio e devo dire essere questa solo una delle tante branche dell'informatica. Una volta approfondito se pur in modo minimo una di queste si sono aperti mondi nuovi ed inesplorati.

In più, cosa non trascurabile, ho imparato ad utilizzare il web. Mi rendo conto che detto così, da un futuro ingegnere informatico, potrebbe sembrare strano. Ma è così. L'utilizzo che prima facevo del

web non era produttivo. Adesso ho imparato a cercare la soluzione migliore in internet, confrontarle e scegliere.
Che tutte le risposte si possano trovare sul web è vero, ma bisogna anche saperle cercare e bisogna anche saper trovare la soluzione giusta al proprio problema.
Ed è proprio questa la cosa più importante che ho imparato: riuscire a trovare una soluzione valida e competitiva da solo, svilupparla e implementarla nei migliori dei modi.
Infine devo dire di aver scoperto il mondo del lavoro e ne sono rimasto, in un certo qual modo, affascinato: certo complice di questa mia sensazione è certamente l'ambiente di lavoro che ho trovato.
Infatti il clima che si respira nell'azienda in cui ho lavorato è molto sereno e sicuramente non stressante. In più ho trovato persone pronte ad aiutarmi quando ne avevo bisogno e, cosa non trascurabile, appoggiarmi e incoraggiarmi a trovare soluzioni nuove e migliori.
Il sistema che ho implementato è stato molto apprezzato e viene tuttora utilizzato per il monitoraggio di tutti i clienti dell'azienda.
Il Nagios implementato continua, infatti, a monitorare i server principali, e in alcuni casi anche le intere reti, dei clienti dell'azienda con successo e adempiendo al meglio il suo compito.

BIBLIOGRAFIA

http://www.nagios.org
Documentazione ufficiale di Nagios, Ethan Galstad

http://www.netsaint.org
Documentazione ufficiale di Netsaint, Ethan Galstad

http://www.hp.com
Sito ufficiale HP

http://www.tivoli.com
Sito ufficiale IBM tivoli-netview

http://www.falcom.de
Sito ufficiale della Falcom produttrice di Modem GSM

http://smslink.sourceforge.net/
Sito ufficiale del progetto smslink

Contatti:
Matteo Vitolo
matteo@vitolo.info
http://www.vitolo.info