Estudo de uma Ferramenta de Gestão de Redes

SUMÁRIO

Este trabalho é orientado a estudar uma ferramenta de gestão de redes, o Nagios. Apresenta-se o caminho percorrido duma instalação com sucesso, numa plataforma FreeBSD 5.1 i386. São expostas, duma forma crítica, as soluções encontradas para os problemas. Refere-se e explica-se na sua generalidade, as potencialidades desta ferramenta. É feita, na óptica da aplicação, uma avaliação construtiva e é estudado um caso simples e real de aplicação desta. O foco deste estudo, contudo, visa aplicar-se à visão de utilidade e escalabilidade do Nagios, bem como as suas vantagens e desvantagens inseridas no modo como foi concebida e idealizada.

Índice

Capítulo 1 – Introdução

"System and network monitoring is essential for systems administrators (…) give Nagios a try. You will not be disappointed."

Syed Ali em *Sys Admin, Outubro 2003*

"nagios-1.0_1 - Extremely powerful network monitoring system."

FreeBSD, package information

O Nagios é uma ferramenta GPL escrita por Ethan Galstad que pode ser utilizada para monitorizar nós numa rede. Como base a uma avaliação fidedigna, vai-se criar um ambiente de estudo real para o teste desta ferramenta. Temos como ponto de partida um *router/servidor* Pentium 200 MMX em FreeBSD 5.1 localizado numa LAN particular. O baixo processamento desta máquina é-nos vantajoso para avaliar o peso computacional, bem como a complexidade do Nagios. A rede de teste tem a seguinte topologia:

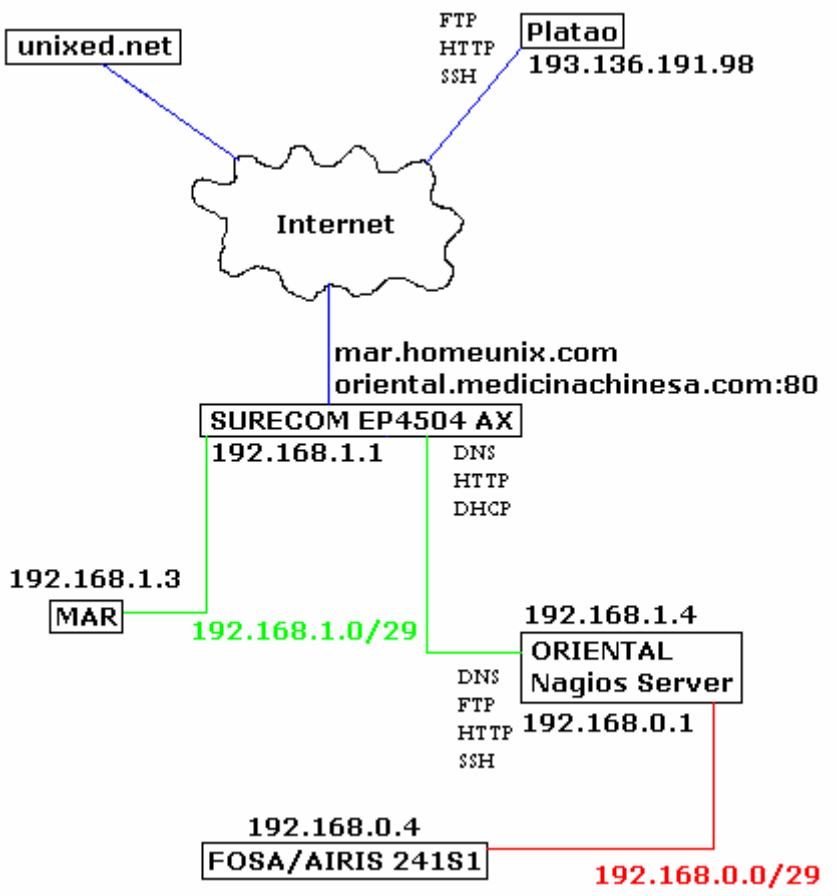


Figura 1 – Topologia da rede de teste.

O *router/server* é a máquina ORIENTAL na qual vai ser instalado o Nagios. Os serviços que correm nesta são acessíveis pela *Internet* devido ao *port forwarding* efectuado no *router* SURECOM.

Primeiramente, no segundo capítulo vamos desvendar brevemente o que é a licença GPL. Ao avançarmos para o terceiro capítulo vamos abordar a instalação passo a passo do Nagios 1.1, deixando a configuração ser tratada no quarto capítulo. A crítica como avaliação geral dos últimos dois passos é feita no quinto

capítulo. No sexto capítulo é tratada a motivação, onde vão ser apresentadas dum modo geral as capacidades deste. Todo a informação técnica detalhada e necessária para a compreensão do funcionamento da ferramenta é fornecida no sétimo capítulo. De seguida, no oitavo capítulo, vamos fazer testes a serem analisados posteriormente no nono capítulo. Como ponto intermédio de desenvolvimento é feita uma crítica construtiva no décimo capítulo. Para o décimo primeiro capítulo deixamos os requisitos e noções avançadas de Nagios. Finalmente no décimo segundo capítulo, antes da conclusão, são referidos os novos desenvolvimentos para esta ferramenta. A componente prática deste trabalho dedica-se exclusivamente às potencialidades de monitorização da ferramenta, no entanto a componente de gestão é abrangida na parte teórica.

O objectivo deste trabalho é oferecer uma visão íntegra das potencialidades que a ferramenta Nagios possibilita a um administrador de redes e sistemas. Não existe a intenção de fazer deste trabalho um manual detalhado das funcionalidades de Nagios, embora nalguns aspectos o nível de detalhe seja muito elevado.

Capítulo 2 – O que significa *GPL*?

GPL é o acrónimo de *General Public License*. A licença mais difundida sobre GPL é a GNU GPL, base do projecto da *GNU*. As licenças para a maior parte do *software* são concebidas para privar-nos de partilhar e alterar. Em contraste, a *GNU General Public License* tem a intenção de garantir a nossa liberdade de partilhar e alterar *software* gratuito - garantir que o *software* é gratuito para todos os utilizadores. O projecto GNU representa um esforço colaborativo guiado pela *Free Software Foundation*, com o objectivo de produzir um conjunto de ferramentas de *software* de alta qualidade, sobre a *GPL*. Programas produzidos pelo projecto *GNU* incluem o editor de texto Emacs e o compilador *GNU C Compiler* (*GCC*) que, são largamente usados não só por plataformas grátis mas, também em várias plataformas proprietárias.

Capítulo 3 – Instalação do Nagios

Quem quiser instalar Nagios deve ter o seguinte aviso em mente...

“Installing and configuring Nagios is rather involved. You can't just compile the binaries, run the program and sit back. There's a lot to setup before you can actually start monitoring anything. Relax, take your time and read all the documentation - you're going to need it.”

Ethan Galstad, Nagios Version 1.0 Documentation 2002

Nagios consiste num programa central e alguns *plugins*. O programa central invoca os *plugins* para fazerem a monitorização. Este é escrito em *C* e alguns dos *plugins* são escritos em *C* e *Perl*. O suporte de instalação é dedicada a Linux, embora este seja compatível com todos os *POSIX compliant*. Nisto inclui-se o FreeBSD 5.1, no qual vamos instalar a ferramenta.

Inicialmente vamos tentar instalar a Versão 1.1.

Começamos por fazer o download do *Nagios Version 1.1* em http://www.nagios.org/download, obtendo o ficheiro *nagios-1.1.tar.gz*. A seguir fazemos o *download* do conjunto básico de plugins *nagios-plugins-*

1.3.1.tar.gz em http://sourceforge.net/project/showfiles.php?group_id=29880. Nesta última referência podem-se encontrar disponíveis muitos outros *plugins* para *download*.

A seguir é melhor criar o utilizador "nagios" e o grupo "nagios". Pode-se fazer isto com os seguinte comando:

```
[root@oriental local] # adduser
```

Depois vamos descomprimir o ficheiro do nagios com o seguinte comando:

```
[root@oriental local] # tar xvfz  nagios-1.1.tar.gz
```

Fazer `cd` para o directório nagios-1.1 e correr o comando de configuração:

```
[root@oriental nagios-1.1] # ./configure
```

O comando *configure* tem os seguintes *defaults* os quais não devem ser alterados:

```
[root@oriental nagios-1.1] # ./configure --prefix=/usr/local/nagios
  --with-cgiurl=/nagios/cgi-bin --with-htmurl=/nagios
  --with-nagios-user=nagios --with-nagios-grp=nagios
```

Compilar o Nagios:

```
[root@oriental nagios-1.1] # make all
```

Instalar os binários, os *CGIs* e os ficheiros *HTML*:

```
[root@oriental nagios-1.1] # make install
```

Instalar o *script* de inicialização em `/usr/local/etc/rc.d/nagios/` :

```
[root@oriental nagios-1.1] # make install-init
```

Instalar e configurar as permissões do directório para suportar o ficheiro de comandos externos:

```
[root@oriental nagios-1.1] # make install-commandmode
```

Não esquecer de alterar as permissões do directório `/var/spool/nagios/rw/` onde vai correr o ficheiro nagios.cmd, o qual é usado pelos CGIs do Nagios. Mudar o grupo desse directório para o mesmo grupo do *user* que corre o *Apache* ou outro *HTTP Server*. No nosso caso mudámos para `www` com o comando `chgrp`.

Instalar os ficheiros de exemplo para configurações:

```
[root@oriental nagios-1.1] # make install-config
```

De notar que é necessário alterar estes ficheiros antes de se poder correr o Nagios (assunto deixado para o próximo capítulo).

Fazer `cd` para o directório `/usr/local/nagios`.

Nesse directório deve ver cinco subdirectórios diferentes. Na tabela seguinte é dada um breve descrição do que cada directório contém:

Sub-Directório	Conteúdo
bin/	Programa Central do Nagios.
etc/	Os ficheiros de configuração Main, resource, object e CGI devem ser postos aqui.
sbin/	CGIs.
share/	Ficheiros HTML (para interface *Web* e documentação *online*).
var/	Directório vazio para os ficheiros log.

Tabela 1 – Subdirectórios de /usr/local/nagios – LINUX

Para o Nagios ter qualquer utilidade é necessário instalar os *plugins*. Os *plugins* são *scripts* ou binários que fazem todas as verificações utilizadas para monitorizar sistemas e serviços.

Vamos primeiro descomprimir o ficheiro dos plugins com o seguinte comando:

```
[root@oriental local] # tar xvfz  nagios-plugins-1.3.1.tar.gz
```

Fazer `cd` para o directório `nagios-plugins-1.3.1` e correr o comando de configuração:

```
[root@oriental nagios-plugins-1.3.1] # ./configure
```

Como escolhemos o caminho por defeito par a instalação do Nagios (`/usr/local/nagios`), não é necessário fornecer qualquer parâmetro. No caso de querermos alterar o caminho de destino, devemos correr o comando com os seguintes dados:

```
[root@oriental nagios-plugins-1.3.1] # ./configure
 --prefix=BASEDIRECTORY --with-nagios-user=SOMEUSER
 --with-nagios-group=SOMEGROUP --with-cgiurl=SOMEURL
```

Compilar os *plugins*:

```
[root@oriental nagios-plugins-1.3.1] # make all
```

Finalmente instalar os binários dos *plugins*:

```
[root@oriental nagios-plugins-1.3.1] # make install
```

Na realidade, não nos foi possível concretizar através deste método a instalação correcta dos plugins. Ao compilar são detectadas dependências não assimiladas na configuração. Embora a instalação anterior tenha problemas em FreeBSD, não apresenta quaisquer problemas se for feita numa distribuição de Linux. Isto leva-nos a outra opção de instalação do Nagios, que é fazer o download directo da página do FreeBSD, com o comando `sysinstall` disponível em `/usr/sbin/`:

```
[root@oriental nagios-plugins-1.3.1] # sysinstall
```

Escolhendo as seguintes opções:

```
Configure > Packages > FTP > Primary Site
```

Escolher os seguintes pacotes de instalação:

```
nagios-1.0_1
nagios-plugins-1.3.0
```

A árvore de FreeBSD usa *standards* diferentes do Linux, o que implica uma tabela diferente, não centralizada, dos componentes do Nagios. Verifica-se agora uma instalação incorporada na gestão global de directórios do FreeBSD. Isto é mostrado na tabela seguinte:

Directório / Ficheiro	Conteúdo
/usr/local/bin/nagios	Programa central do Nagios.
/usr/local/etc/nagios/	Os ficheiros de configuração Main, resource, object e CGI.
/usr/local/share/nagios/	Ficheiros HTML.
/usr/local/libexec/nagios/	Directório dos plugins do Nagios.
/usr/local/etc/rc.d/nagios	Script de inicialização do Nagios.
/usr/local/sbin/	CGIs.
/var/spool/nagios/rw/nagios.cmd	Ficheiro usado pelos CGIs para controlar o Nagios.
/var/spool/nagios/nagios.lock	Ficheiro que guarda o PID do Nagios.
/var/spool/nagios/archives/	Directório para os ficheiros log.

Table 2 - Directórios do Nagios em FreeBSD

Após esta instalação e por pesquisa em fóruns sobre o assunto, foi-nos dado a conhecer que é possível fazer uma compilação de todos os ficheiros fonte do Nagios, sem ser necessário alterar código. Para isto é necessário usar o *gmake* em vez do *make*.

Capítulo 4 – Configuração do Nagios

A configuração do Nagios é uma tarefa que tem de ser muito bem pensada, especialmente se falamos de redes muito complexas. É possível aprender a configurar a nossa própria rede apenas reparando nos ficheiros exemplificativos disponíveis no directório de configuração, nomeadamente `/usr/local/etc/nagios/` (FreeBSD) ou `/usr/local/nagios/etc/` (Linux):

[root@oriental nagios] # ls

cgi.cfg-sample
checkcommands.cfg-sample
contactgroups.cfg-sample
contacts.cfg-sample
dependencies.cfg-sample
escalations.cfg-sample
hostextinfo.cfg-sample
hostgroups.cfg-sample
hosts.cfg-sample
misccommands.cfg-sample
nagios.cfg-sample
resource.cfg-sample
serviceextinfo.cfg-sample
services.cfg-sample
timeperiods.cfg-sample

Notar que os ficheiros contêm o sufixo "–*sample*" porque são exemplos de ficheiros de configuração. Para usar efectivamente estes ficheiros é necessário retirar "-*sample*".

Os ficheiros de configuração podem ser divididos em cinco tipos:
1. Configuração do ficheiro principal (*nagios.cfg*)
2. Ficheiros de *resource* (*resource.cfg*)
3. Ficheiros de configuração de objectos

4. Ficheiro de configuração de CGI (cgi.cfg)
5. Ficheiros de configuração para informação estendida

Nagios é altamente costumizável e por isso só vamos abarcar algumas das directivas de configuração.

O ficheiro de configuração principal contém um número de directivas que afectam o modo como o Nagios opera. Este ficheiro é lido pelo processo do Nagios e pelos CGIs. Este é o primeiro ficheiro de configuração a editar.

Os ficheiros *resource* podem ser usados para definir *macros*, além de poderem conter outras informações (como configurações de ligações de base de dados). Isto depende do modo como o Nagios foi compilado. O propósito de ter ficheiros *resource* é de guardar informações importantes e de não as disponibilizar aos CGIs. É possível especificar um ou mais ficheiros *resource* usando a directiva *resource_file* no ficheiro principal de configuração. Macros são variáveis usadas nas definições de comandos, as quais são substituídas pelos seus valores na altura em que estes são executados. Isto permite definir comandos genéricos que preencham todo o tipo de necessidades.

Os ficheiros de configuração de objectos (por motivos históricos chamados de ficheiros de configuração de *hosts*) são usados para definir *hosts*, serviços, grupos de *hosts*, contactos, grupos de contactos, comandos, etc. É aqui que se define que coisas nós queremos monitorizar e como fazê-lo. De acordo com Ethan Galstad (o colaborador principal do Nagios), um objecto é simplesmente um termo genérico que este usa para descrever várias definições de dados necessários para monitorizar qualquer coisa. Os ficheiros que contêm definições de objectos são constituídos pelos seguintes:

1. checkcommands.cfg
2. contactgroups.cfg
3. contacts.cfg
4. dependencies.cfg
5. escalations.cfg
6. hostgroups.cfg
7. hosts.cfg
8. misccommands.cfg
9. services.cfg
10. timeperiods.cfg

O ficheiro contacts.cfg contém informação acerca de contactos. Um contacto é um administrador de sistemas ou qualquer outro tipo de pessoa que vai ser notificada na eventualidade duma emergência. A forma de notificação pode ir desde um *e-mail* até um *SMS* para o telemóvel.

Grupos de contactos (contactgroups.cfg) são muito úteis porque é possível criar grupos, por exemplo, de AdministradoresHTTP ou AdministradoresSMTP e usar estes grupos para serem notificados dos respectivos serviços.

É possível ao configurar correctamente o ficheiro timeperiods.cfg, definir horários de notificações. Isto é muito útil se queremos, por exemplo, não incomodar com notificações de emergência certos trabalhadores que fazem folga aos fins-de-semana e avisar outros que estão de serviço.

Para definir um qualquer *host* que o Nagios vai monitorizar é necessário configurar o ficheiro hosts.cfg. Definir um *host* implica fornecer um nome, um endereço IP e definir outros parâmetros que o Nagios vai utilizar.

A configuração de ficheiros de objectos podem ser feitas através método "antigo" ou através do método baseado em *templates*. O método "antigo" é baseado apenas em ficheiros mas não usava *templates*. Ao usar um *template* na definição dum serviço ou objecto, simplifica-se a adição de novos *hosts* e serviços para monitorização. Por exemplo, considera-se um novo *host* ao editar o ficheiro hosts.cfg. Um *host*

monitorizado é uma instância duma interface *host*, definindo-lhe os atributos acerca do próprio *host* (como o nome, o endereço IP, quantas verificações são feitas pelo Nagios).

O ficheiro hostgroups.cfg deixa-nos criar um grupo lógico de *hosts*. O agrupamento lógico de servidores com base no serviço a monitorizar é um método possível. Um *host* deve pertencer a pelo menos um grupo e pode pertencer a múltiplos grupos.

No ficheiro services.cfg podem-se definir serviços para monitorizar os nossos *hosts* previamente definidos. Podem-se usar *templates* na definição de serviços, tal como na definição de *hosts*. É possível criar um serviço para um grupo de servidores POP3 que, herda propriedades do serviço genérico e adiciona as suas próprias propriedades.

O ficheiro dependencies.cfg permite criar dependências entre serviços ou entre *hosts.* Para explicar isto podemos reparar na rede de teste (figura 1). No caso do *router* SURECOM falhar, não é obrigatoriamente necessário notificar o responsável pela máquina Platao, ao detectar obviamente a falha dos serviços deste. Isto permite no fundo adicionar alguma inteligência artificial ao Nagios.

O ficheiro de configuração de *CGI* contém um número de directivas que afectam a operação dos *CGIs*.

Os ficheiros de configuração para informação estendida são usados para definir informação adicional para *hosts* e serviços que devem ser usados pelo CGI. É aqui que se definem coisas como as coordenadas de desenho, imagens dos *hosts*, informações acerca de *hosts* ou serviços, etc.

Os *plugins*, como já foi referido, é que fazem a monitorização. O programa principal do Nagios invoca os *plugins* para verificarem os *hosts* e serviços. O Nagios disponibiliza alguns *plugins* mas, é possível escrever os nossos próprios *plugins* se queremos monitorizar qualquer *host* ou serviço. Os *plugins* que vêm com o Nagios têm ajuda disponível quando executados com a opção –h.

Quando especificamos uma opção check_command no ficheiro services.cfg para um serviço em particular, o Nagios procura esse comando de verificação no ficheiro checkcommands.cfg e corre o comando com base nas opções especificadas.

O Nagios fornece uma interface para *Web*. Se quiseres utilizar o Nagios com interface *Web*, é necessário editar o ficheiro cgi.cfg e também o ficheiro de configuração do *Web server*. No nosso caso usamos o Apache e as alterações necessárias são as seguintes:

```
ScriptAlias /nagios/cgi-bin /usr/local/sbin/
     <Directory "/usr/local/sbin">
         AllowOverride AuthConfig
         Options ExecCGI
         Order allow,deny
         Allow from all
     </Directory>

     Alias /nagios/ /usr/local/share/nagios/
     <Directory "/usr/local/share/nagios">
         AllowOverride AuthConfig
         Options None
         Order allow,deny
         Allow from all
     </Directory>
```

Para além destas alterações também configurámos o Apache com permissões de acesso à página do Nagios. Utilizámos o modo *Basic* e não o *Digest* devido a conflitos com os *Web browsers*. No caso do leitor efectuar estas alterações, tenha em atenção a configuração do ficheiro cgi.cfg, nomeadamente em relação às configurações de permissões.

É feita uma listagem de todos os ficheiros de configuração no Anexo A.

Capítulo 5 – Crítica à Instalação e Configuração

Primeiro é de apontar um ponto negativo nos *scripts* de configuração do Nagios aquando da compilação. O suporte é dedicado a distribuições Linux, o que poderia ser melhorado com *scripts* que respeitassem todas os outros *POSIX compliant*. Apesar disto, a única consequência que trás é obrigar o utilizador a reconfigurar os *scritps* de configuração.

A instalação de Nagios num dado sistema não é acessível a qualquer utilizador, aliás, sem conhecimentos de redes e das plataformas visadas é aconselhável que contrate profissionais para fazerem o trabalho. Dito isto é necessário também referir que mesmo para quem tenha conhecimentos da área, uma adaptação ao Nagios implica a leitura de muita documentação. As configurações e detalhes mostrados no capítulo de configuração são apenas amostras do que se é capaz de fazer ao preparar a gestão dum sistema. Não é o propósito deste trabalho incluir toda essa informação e nem é plausível isso acontecer, já que "assemelha-se bastante" a documentar todas as jogadas possíveis num jogo de xadrez.

O que se conclui ao instalar o Nagios num sistema é que este *software* tem muitas vantagens neste campo. Não é necessário ter uma máquina dedicada só para fazer a monitorização e também não é obrigatório instalar *software* adicional nos *hosts* que queremos monitorizar.

Esta ferramenta é muito versátil, já que a sua aplicação em sistemas *POSIX compliants* implica uma população alvo muito grande. Na realidade isto é apenas referente ao Servidor do Nagios, porque os clientes activos ou passivos existem para outros sistemas, nomeadamente para distribuições de Windows.

É também importante dizer que através da configuração do Nagios, verifica-se que este está bastante vocacionado para empresas, sendo muito funcional neste campo. É possível definir horários de trabalho para notificações, ou outro tipo de horários, bem como definir exactamente quem é responsável pelo quê e que acessos tem na gestão do sistema.

Capítulo 6 - Motivação

O Nagios é uma aplicação orientada à monitorização de sistemas e redes. Verifica os sistemas e redes que são especificados, alertando quando as situações correm mal e quando as situações melhoram.

Originalmente o Nagios foi desenhado para correr em sistemas Linux, embora deva correr em todas as plataformas *UNIX*.

O Nagios é muitíssimo versátil e flexível, contendo muitas funcionalidades. Algumas dessas funcionalidades são:

monitorização de serviços de rede (SMTP, POP3, HTTP, NNTP, PING, SSH, etc.);
monitorização de recursos de *hosts* (carga de processamento, utilização de disco, etc);

desenho simples de *plugins* que permite aos utilizadores desenvolverem os seus próprios verificadores de serviços;
verificadores de serviços em paralelo;
habilidade de definir hierarquia entre *hosts* da rede ao usar *parent hosts*, permitindo a detecção e distinção entre *hosts* que estão em baixo e aqueles que não são alcançáveis;
notificar quando ocorrem problemas ou resoluções de serviços (via *e-mail*, *pager*, ou por um método definido pelo próprio utilizador);
habilidade de definir tratadores de eventos para serem corridos durante os serviços ou eventos de *hosts* para resolução pró-activa de problemas;
rotação automática de *logs*;
suporte para a implementação de monitorização redundante por *hosts*;
interface *Web* para visionar o estado corrente da rede, as notificações e o historial de problemas, ficheiros *log*, etc.;

Capítulo 7 – Noções Intrínsecas

Nagios Remote *Plugin* Executer(NRPE)

Este *software* tem como objectivo correr *plugins* locais de *hosts* remotos a pedido do Nagios. O *plugin* **check_nrpe** corre no *host* Nagios e envia pedidos de execução de *plugins* ao agente **nrpe** a correr no *host* remoto. O agente executa então o *plugin* requisitado e devolve o *output* do *plugin* para o **check_nrpe** do *host* Nagios acompanhado do código de retorno. O **check_nrpe** passa então os elementos para o Nagios tal como se fossem seus. Consegue-se assim uma relativa transparência em relação ao facto de os *plugins* estarem a ser executados em *hosts* remotos.
O agente **nrpe** pode correr tanto como um *daemon* independente como um serviço sob **inetd**.

Nagios Service Check Acceptor (NCSA)

Este software permite que um *host* envie mensagens de verificação de serviço ao *host* Nagios por sua própria iniciativa, ou seja, permite que o *host* Nagios receba informação sobre disponibilidade de serviços de uma forma passiva. O cliente **ncsa** pode tanto correr como uma aplicação independente num *host* monitorizado ou fazer parte integrante dum servidor Nagios. Esta última solução é utilizada para formar ambientes de monitorização distribuída utilizando comandos **ocsp**.

Serviços implementados em *hosts* em baixo

Sempre que é recebida como resposta de uma verificação de serviço uma mensagem de serviço indisponível, o Nagios tenta descobrir se o *host* em que o serviço está implementado está operacional, enviando "*ping*s" para o mesmo e verificando se é obtida alguma resposta. Em caso afirmativo conclui-se que o *host* está operacional e que o problema é do serviço. Caso contrário, o Nagios dá o *host* como inoperacional e silencia todas as notificações de alerta de indisponibilidade do serviço.

Hosts locais

Designamos por *hosts* locais aqueles que estão situados no mesmo segmento de rede em que o *host* está a correr o Nagios não existindo *routers* ou *firewalls* entre eles.
Como estes *hosts* estão, em termos de hierarquia de rede, no mesmo nível que o *host* Nagios, não possuem nós "pais", logo a opção <*parent_hosts*> deve ser deixada em branco.

A monitorização de *hosts* locais é bem mais simples do que a de *hosts* remotos (vista mais à frente). O Nagios precisa apenas de executar o comando de verificação desse *host* para determinar se ele está, ou não, operacional.

Hosts remotos

Hosts remotos são aqueles que se situam num segmento de rede diferente daquele onde se encontra o *host* Nagios. O campo <parent_*host*> permite definir a hierarquia de rede dos *hosts* remotos especificados. Ele deve conter os nomes dos *hosts* imediatamente acima hierarquicamente do *host* em questão.

A monitorização de *hosts* remotos é um pouco mais complicada do que a de *hosts* locais. È preciso destinguir casos em que o *host* está em baixo de casos em que está inacessível. Para tal, sempre que é recebida, como resposta dum comando de verificação, uma mensagem que dá o *host* como inoperacional, o Nagios percorre a hierarquia no sentido ascendente até encontrar um *host* operacional. Se isto se verificar no nível imediatamente acima do *host* monitorizado então conclui-se que o *host* está em baixo. Caso contrário, deduz-se que são os acessos ao *host* que estão em baixo e que estão a tornar o *host* inacessível.

Falhas de rede

O Nagios inclui um CGI (*outage* CGI) para assistir o utilizador na identificação das causas de falhas de rede. Este CGI é especialmente útil para redes de maior dimensão, pois ajuda os administradores a isolar e resolver os problemas que estão a causar mais impacto na rede com maior rapidez.

Convém notar que o CGI não vai identificar as causas exactas da falha de rede mas antes identificar os *hosts* que estão a causar mais problemas. O aprofundamento na identificação do problema é uma tarefa deixada para o utilizador.

Para um *host* ser dado como causador de problemas tem de verificar duas condições:

- Tem de estar em baixo ou inacessível e, pelo menos um dos seus *hosts* "pais" (imediatamente acima hierarquicamente) tem de estar operacional.
- Todos os seus *hosts* "filhos"(imediatamente abaixo hierarquicamente) têm de estar em baixo ou inacessíveis e não ter nenhum *host* "pai" operacional.

Se, e só se, ambas estas condições forem verificadas é que o *host* é dado como causador de problemas.

Para além de identificar os *hosts* causadores de problemas o *outage CGI* determina também o número de *hosts* que estão a ser afectados pela falha de cada um desses *hosts*.
Tirando proveito deste cálculo e determinando ainda o número de serviços que estão a ser indisponibilizados pela falha do *host* o Nagios pode calcular o grau de severidade do efeito que esta falha está a provocar na rede. Para este cálculo o número de *hosts* tem um peso quatro vezes maior que o número de serviços. Este valor é então utilizado para ordenar os *hosts* causadores de problemas por severidade do problema causado.

Notificações

As notificações servem para fornecer informações em tempo real sobre a operacionalidade de *hosts* ou serviços para as pessoas visadas.

Uma notificação ocorre nas seguintes situações:

- Instantes de mudança de estado *hard*
- Quando um *host* ou um serviço se mantém num estado *hard* inoperacional e o tempo passado desde a última notificação é maior do que o tempo definido no campo `<notification_interval>`. Caso o utilizador não queira receber notificações nesta situação, basta estabelecer o intervalo a 0.

Cada serviço possui um campo `<contact_groups>` que define os grupos de contactos a serem notificados de informações referentes àquele serviço. Mesmo que um contacto pertença a dois ou mais grupos de contacto referidos neste campo o Nagios não irá enviar notificações duplicadas a esse contacto.
Cada grupo de *hosts* inclui também um campo `<contact_groups>` com uma função análoga à anterior. Aqui definem-se o grupo de contactos a serem notificados sobre informações referentes àquele grupo de *host*s. De novo nunca são enviadas notificações em duplicado para os contactos.

Podem ser definidos três tipos de filtros sobre as notificações a enviar. As notificações são filtradas pela ordem apresentada:

Filtros no âmbito do programa:

Verificam se, de acordo com a configuração geral do Nagios, as notificações devem ser enviadas. Esta configuração tem um valor inicial, definido na directiva `enable_notifications` e pode ser alterado a qualquer momento através do interface *web*. Este filtro não permite fazer qualquer tipo de selecção sobre quais as notificações que são passadas para o próximo filtro e quais as que ficam retidas. Ou passam todas ou não passa nenhuma.

Filtros de serviço e de *host*:

Estes filtros permitem reter notificações segundo critérios relacionados com os serviços ou *hosts*. Nomeadamente:

- Notificações de *host* ou serviço inoperacional que se inserem dentro do período de *downtime* (período no qual é suposto o *host* ou serviço estarem indisponíveis) desse mesmo *host* ou serviço.
- Notificações de *host* ou serviço inoperacional quando estes estão em *flapping* (a mudar de estado operacional para inoperacional de uma forma intermitente). Esta filtragem implica que esteja activa a detecção de *flapping*.
- Notificações que não estão de acordo com os critérios específicos do serviço ou *host*. Estes critérios são definidos num conjunto de opções incluídas na definição do *host* ou do serviço e definem se devem ser enviadas notificações quando o *host* vai abaixo, fica inacessível, etc.
- Notificações que não se inserem dentro do período de notificação definido no campo `<notification_period>` da definição do *host* ou serviço, ou notificações cujo tempo passado desde a última notificação relativa ao mesmo estado seja menor que o tempo definido no campo `<notification_interval>` da definição do *host* ou serviço.

Filtros de contactos

As últimas filtragens efectuadas às notificações estão relacionadas com critérios específicos de cada contacto.

São filtradas quaisquer notificações que não estejam de acordo com as opções incluídas na definição do contacto. Podem ser filtradas por exemplo, notificações que se resumam a entradas em estado de alerta, ou que se situem fora do período de tempo definido no campo `<notification_period>` da definição do contacto.

Plugins

Os *plugins* são uma parte fundamental da arquitectura do Nagios. Como o processo central do Nagios não inclui qualquer tipo de mecanismo para verificação de *host*s ou serviços, todo o software para esta tarefa está inserido nos *plugins*. Assim os *plugins* servem de intermédio entre o processo Nagios e o *host* ou serviço monitorizado. Para fazer uma verificação, o Nagios manda executar o *plugin* definido para verificar esse serviço, o *plugin* toma as acções necessárias para a verificação (enviar um *ping* a um *host* por exemplo), e devolve o resultado ao Nagios para o processamento posterior (Tratadores de eventos, etc).
Esta arquitectura tem a vantagem de ser extremamente versátil. Basta desenvolver um *plugin* que efectue uma acção de teste para um dado equipamento ou serviço que o Nagios pode monitorizá-lo. Esta versatilidade tem porém um ponto negativo. Se quisermos fazer uma monitorização não só de operacionalidade mas também de outros factores(temperatura, carga de processamento), e exactamente por o processo central do Nagios ser tão versátil, há uma dificuldade em tratar a informação enviada de resposta pelo *plugin* de uma forma específica para esse tipo de informação, por exemplo, em termos de visualização (gráficos, animação, etc.).

Marcação de verificações

É possível ajustar três parâmetros em termos de tempos de marcação de verificações de serviços. Em primeiro lugar temos o intervalo normal entre verificações de serviço que define o intervalo entre verificações enquanto o serviço está disponível. Em segundo lugar o intervalo de repetição de verificações estipula o intervalo de tempo entre verificações quando o serviço passa ao estado indisponível *soft* (ver mais à frente), e por último o período de tempo fora do qual não podem ser feitas verificações de serviço. De referir que todos estes parâmetros são específicos do serviço.

A marcação das verificações a serviços ou *host*s é feito de forma a, por um lado, tentar equilibrar o processamento na máquina onde está a correr o Nagios, e por outro lado, minimizar a carga nos *hosts* remotos.
Para realizar a primeira tarefa o Nagios distribui as verificações iniciais a serviços ao longo do intervalo de tempo necessário para a verificação inicial a todos os serviços. O valor dado ao espaçamento entre verificações é então o valor médio dos intervalos normais entre verificações. Dizemos verificações iniciais porque, passado algum tempo, o marcação das verificações a serviços torna-se bastante caótico devido aos diferentes intervalos normais entre verificações dados a cada um dos serviços. Porém o Nagios tenta, pelo menos na fase inicial, manter as verificações relativamente equidistantes.
Quanto à tarefa de minimizar a carga nos *host*s remotos, o Nagios tenta não fazer as verificações de todos os serviços disponibilizados por um dado *host* consecutivamente. O percorrer dos serviços é antes feito "saltando" um certo número fixo de serviços de modo a, em média, depois de cada salto, ser efectuada uma verificação a um serviço do *host* seguinte. Assim o número de *hosts* a saltar é determinado pelo número médio de serviços por *host*. Este processo é designado por intercalação.

Verificações de *hosts*

Ao contrário das verificações de serviços, as verificações de *hosts* não são marcadas de uma forma regular. São antes executadas quando o Nagios vê necessidade nisso.
Sempre que uma verificação de serviço dá uma reposta de serviço indisponível é efectuada uma verificação do *host* onde esse serviço está implementado. Caso também esta dê uma resposta negativa o *host* passa a um estado inoperacional *soft* e o Nagios continua a executar ininterruptamente verificações ao *host* até que,

ou é recebida uma resposta positiva, ou é atingido o número máximo de verificações e o *host* passa a um estado inoperacional *hard*.

Estados *soft* e *hard*

Um serviço ou *host* atinge um estado *soft* de erro quando é recebida em resposta de uma verificação uma mensagem de inoperacionalidade. Um estado *soft* de recuperação é atingido quando é recebida em resposta de uma verificação uma mensagem de operacionalidade enquanto o serviço ou *host* se encontrava em estado *soft* de erro.
Sempre que há uma alteração de estado *soft*, é acrescentado um registo ao *log* se essa opção tiver sido accionada aquando da configuração. São também invocados todos os tratadores de eventos definidos pelo administrador para lidar com a mudança de estado. Não são enviadas quaisquer notificações, pois uma mudança de estado suave não é considerada uma mudança no estado real do *host* ou serviço.

Por outro lado, um serviço ou *host* atinge um estado *hard* de erro quando são recebidas consecutivamente um número pré-definido de mensagens de inoperacionalidade em resposta de verificações. Um estado *hard* de recuperação é atingido quando é recebida em resposta de uma verificação uma mensagem de operacionalidade enquanto o serviço ou *host* se encontrava em estado *hard* de erro.
Sempre que há uma alteração de estado *hard*, também é acrescentado um registo ao *log* se essa opção tiver sido accionada aquando da configuração. São invocados todos os tratadores de eventos definidos pelo administrador para lidar com a mudança de estado.
São, desta vez, enviadas notificações (se o sistema de filtragem de notificações o permitir) aos contactos incluídos no grupo de contactos do servidor ou *host*.
Se o *host* ou servidor se mantiver num estado *hard* de erro por um intervalo de tempo maior do que o definido para o intervalo entre notificações são de novo enviadas notificações.

Capítulo 8 – Testes

Neste capítulo vamos elaborar alguns testes à rede de teste. É necessário para isto referir o estado da rede e dos seus sistemas. A máquina Platao encontra-se por detrás duma *firewall* com uma filtragem a todos os pacotes ICMP. Os *hosts* 241S1, MAR e SURECOM WIRELESS estão desligados. O *router* SURECOM tem um serviço HTTP na porta 12345, ao invés da porta 80. O *router/server* ORIENTAL tem os seus serviços disponíveis pelo exterior graças a um *port forwarding* feito no SURECOM para as suas portas.

Pelas imagens (Anexo B) produzidas no interface *Web* do Nagios, concluímos os seguintes estados:

ORIENTAL UNDER SERVERS (oriental-servers)

Host	Services
241s1	PING
mar	PING
oriental	/dev/ad0s1e Free Space FTP HTTP SSH System Load
surecom	PING SURECOM HTTP

Tabela 3

PLATAO Server (platao-servers)

Host	Services
platao	FTP HTTP SSH

Tabela 4

SURECOM WIRELESS UNDER SERVERS (surecomwireless-servers)

Host	Services
surecomwireless	PING

Tabela 5

Cor	-	Estado
Verde	-	*OK*
Amarelo	-	*Warning*
Vermelho	-	*Critical*

Capítulo 9 – Análise de Testes

Com base nos estados acima descritos e com a informação extraída para o Anexo B, deduzem-se diversos factos, os quais podemos inferir. A máquina Platao está estaticamente num estado *down* porque o Nagios não tem dependências (não foi configurado) entre serviços para verificar que, uma máquina que não responde a PINGS mas tem serviços que estão *OK*, implica que ela não está *down* mas sim, e apenas, com um serviço que está no estado *Critical*. Mesmo que neste caso seja possível resolver com dependência de serviços, não faz muito sentido fazer isto para todos os *hosts*. Basicamente, isto devia estar implementado de raiz no Nagios.

É possível gravar os estados dos serviços quando o Nagios se desliga, bastando para isso activar o modo persistente. É notória na marcação dos serviços que existe uma gestão muito envolvente quanto ao desempenho do sistema, quer no arranque do Nagios, quer num estado corrido. É ainda mais notório a abstracção do utilizador quanto à potência de processamento da máquina, pois trata-se dum Pentiium 200 MMX. É claro que os serviços em reparo são de número muito reduzido para se poderem fazer afirmações mais fortes mas, mesmo assim é de reparar um baixo consumo computacional do Nagios.

Em relação ao interface gráfico podem-se tirar ilações muito positivas. Pode-se mesmo dizer que este interface gráfico compete com as versões comerciais, especialmente no seu grau de detalhe e de versatilidade, quer em desenhos esquemáticos, quer em gráficos de médias ponderadas. É fantástico o toque de representação e animação em 3D, com o VRML. Não se pode deixar de referir que o acesso remoto com esta qualidade torna a manutenção duma rede muito versátil. É de notar os vários níveis de utilizadores com a acesso ao Nagios. O utilizador vmac a ligar-se ao Nagios não consegue observar a rede na sua totalidade, mas apenas os segmentos em que é administrador, os quais lhe competem responsabilidades. Isto só repara a ideia que o Nagios é uma ferramenta muitíssimo funcional e preparada para a monitorização de redes duma forma cooperativa.

É uma alegre surpresa verificar que existe a possibilidade de reconfiguração de marcações de verificações de serviços via *Web*.

Numa atenção cuidada aos testes efectuados pode-se concluir que existe uma grande falta de abstracção do *software* em relação aos níveis das camadas inferiores. Isto é negativo para o utilizador com menos conhecimentos mas, na sua generalidade é bastante positivo, já que para o utilizador experiente permite um excelente conhecimento do estado da sua rede e sistemas constituintes.

Capítulo 10 – Noções Avançadas

Tratadores de eventos

Tratadores de eventos são comandos opcionais que são executados sempre que ocorre uma mudança no estado do *host* ou do servidor

Existem dois tipos de tratadores de eventos. Os tratadores de eventos locais e os tratadores de eventos globais.
Os primeiros são desenvolvidos dentro da definição dum *host* ou serviço e são executados apenas quando há uma mudança de estado especificamente nesse *host*.
Os tratadores de eventos globais são executados quando há uma mudança de estado de qualquer *host* ou servidor, e antes da execução dos tratadores de eventos locais.

Convém notar que um tratador de eventos a ser executado tem as mesmas permissões que o utilizador sob o qual o Nagios está a correr, ou seja, quaisquer comandos do tratador de eventos que exijam um nível de permissões mais alto que o do utilizador não serão executados. Isto pode tornar-se um problema quando queremos incluir no tratador de eventos comandos que exijam permissões de *root*.
Poder-se-ia pensar que a solução passaria por ter o Nagios a correr sempre com privilégio de *root* mas isto traria óbvios problemas de segurança. Assim a solução poderá passar por usar o *sudo*, que nos permite ter uma total liberdade para executar comandos, sem aumentar os privilégios do utilizador sob o qual está a correr o Nagios.

Comandos externos

O Nagios tem a capacidade de executar comandos externos, definidos por outras aplicações, que permitem alterar vários aspectos das suas funcionalidades como, por exemplo, parar todos as verificações de serviço ou adicionar um novo *host*.

Para mandar o Nagios executar um dos comandos externos que disponibiliza, a aplicação deve escrever num ficheiro (o ficheiro de comandos) o nome identificativo do comando, os seus argumentos, e o tempo em que o comando foi inserido. Estes três elementos devem ser colocados segundo um formato pré-definido pelo Nagios e que está exemplificado na sua documentação.

Os comandos não são executados pelo Nagios no instante em que são inseridos pela aplicação exterior. O ficheiro de comandos é antes verificado periodicamente de acordo com um período definido pelo utilizador aquando da configuração, e imediatamente a seguir à execução de um tratador de eventos. Estes instantes de consulta são acrescentados aos instantes periódicos com o objectivo de verificar se houve algum comando inserido como consequência da execução do tratador de eventos.

Verificações de serviços e de *hosts* indirectas e passivas

Monitorização de serviços acessíveis publicamente (http, ftp, etc.) e que não estão protegidos por uma *firewall* é relativamente simples de implementar. Basta definir o *plugin* para verificação. O mesmo se aplica a *hosts* que não estão protegidos por uma *firewall*. Porém, quando pretendemos monitorizar serviços que são privados do *host* monitorizado,(como por exemplo, carga de processamento), ou *hosts* e serviços que estão por trás de uma *firewall*, a solução anterior torna-se inviável. É então necessária a existência de uma aplicação, a correr no *host* remoto, que recolha essa informação localmente e a envie para o *host* Nagios.
O Nagios pode interagir com esta aplicação de uma forma activa, em que a aplicação recolhe e envia informação a pedido do *host* Nagios, ou de uma forma passiva, em que o faz por iniciativa própria. Ao primeiro tipo de verificações dá-se o nome de verificações de serviço indirectas. Ao último, verificações de serviço passivas.
O Nagios vem incluído com aplicações para ambos os casos. Para verificações de serviço directas temos o nrpe, para passivas o ncsa. Ambas estas aplicações vêm descritas com algum promenor mais acima (ver software adicional).
Para além dos casos já referidos as verificações de serviço passivas são ainda úteis para monitorizar serviços que são assíncronos por natureza, e que não podem assim ser eficazmente monitorizados de uma forma activa, pois os instantes de verificação determinados pelo Nagios poderão não ser aqueles que resultem numa monitorização mais eficiente.

Verificações de actualidade de resultados de verificações de serviços

O Nagios incluí uma funcionalidade que permite verificar a actualidade dos resultados das verificações de serviços.
Esta sistema é bastante útil quando temos um serviço cujas verificações são feitas de uma forma passiva, e pretendemos garantir que os resultados que obtivemos dessas verificações não ficam demasiado "velhos", ou seja, pretendemos impor um intervalo de tempo máximo entre verificações. Se esta opção for activada, e estiver definido o intervalo de tempo máximo entre verificações desejado, o Nagios, sempre que detectar que esse intervalo foi ultrapassado, irá forçar uma verificação de serviço activa de modo a renovar o resultado relativo à disponibilidade desse serviço.

Monitorização distribuída

O objectivo da monitorização distribuída é distribuir a carga (memória, CPU, etc.) provocada pela verificação de um grande número de *hosts* e um, várias vezes maior, número de serviços por vários servidores Nagios distribuídos que enviam posteriormente os seus resultados a um servidor Nagios central. O envio dos resultados é feito de uma forma passiva para o servidor central utilizando a aplicação ncsa (abordada anteriormente), e normalmente é este que fica encarregue de enviar notificações, definir tratadores de eventos e receber configurações através do interface *web*. Todos os servidores distribuídos podem estar desprovidos destes elementos e resumir-se ao esqueleto do Nagios mais o cliente ncsa.

A verificação de um serviço passa assim a corresponder à seguinte sequência de acontecimentos:

1. O processo Nagios a correr num servidor distribuído efectua uma verificação activa de um serviço de um *host* remoto
2. A resposta é recebida e o resultado é automaticamente enviado para o cliente ncsa (utilizando o comando ocsp que é automaticamente chamado após a verificação de um serviço).
3. O cliente ncsa envia então o resultado para o *daemon* ncsa localizado no servidor central.
4. O *daemon* ncsa escreve um comando de processamento de resultado no ficheiro de comandos externos, dando o resultado recebido como parâmetro.
5. O processo Nagios a correr no servidor central efectua a sua leitura periódica do ficheiro de comandos externos e executa o comando de processamento de resultado, recebendo assim a informação relativa ao resultado da verificação.

Convém notar que o facto de a transmissão de resultados ser feita de uma forma passiva pode trazer problemas, se, por exemplo, um dos servidores distribuídos vai abaixo. Nesse caso, o servidor central deixaria de receber informação sobre um grande número de *host*s e não teria maneira de saber se deveria ter recebido ou não alguma informação. Este problema pode ser resolvido se utilizarmos a verificação da actualidade de resultados de verificações. A utilização deste sistema permite ao servidor central obter activamente a verificação de um serviço ao próprio *host* remoto onde ele está implementado, como situação excepcional, caso deixe de receber resultados de verificações do servidor distribuído encarregue de verificar esse serviço.

Ainda não foi aqui falado de monitorização distribuída de *host*s, porque o facto de a verificação de *host*s não ser feita a intervalos regulares, mas sim quando se vê necessidade nisso, complica grandemente a sua implementação numa arquitectura distribuída, e, por isso, seguindo o exemplo do próprio Ethan Galstad na documentação do Nagios, decidimos não aprofundar essa questão.

Monitorização redundante

O Nagios oferece a possibilidade de definir servidores de monitorização redundantes, que, embora estejam a correr o Nagios, não estão, em condições normais, a efectuar monitorização. Porém, caso exista uma falha num dos servidores monitores, estes podem tomar o seu lugar sem prejudicar a monitorização.

Detecção e tratamento de *flapping*

Quando um *host* ou serviço muda de estado muito frequentemente diz-se que este está em *flapping*. É um problema que está normalmente associado a defeitos na configuração e que pode originar uma torrente de notificações de erro e de recuperação. O Nagios tem a capacidade de detectar quando isto está a acontecer e de nesse caso mandar suspender as notificações.

Verificações de serviço paralelas

As verificações de serviço, em Nagios, são executadas por processos que podem correr paralelamente uns aos outros. O Nagios não espera pela resposta de uma verificação para enviar outra. Se ela estiver agendada para aquela altura é enviada logo que possível.

Monitorização de *clusters* de serviços e de *hosts*

É possível monitorizar conjuntos de *hosts* e de serviços como uma só entidade monitorizada.
Tomemos como exemplo um conjunto de *hosts*, ou *cluster*, com um dado serviço implementado mas em que só um deles é que está de facto a disponibilizá-lo para o exterior e os outros estão lá para assegurar que caso haja problemas com esse servidor, eles possam tomar o seu lugar e assegurar a disponibilidade do serviço. Existem vantagens óbvias em interpretar este conjunto de *host*s como uma única entidade e apenas saber sobre a operacionalidade conjunta do *cluster*.

Perseguição de estado

Por defeito, o Nagios não guarda informação no log de qualquer alteração na resposta da verificação que não resulte uma alteração de estado. É, porém, possível activar esta opção. Embora não seja muito comum utilizar esta funcionalidade, ela pode-se tornar interessante se quisermos, por exemplo, vir a consultar os ficheiros de log mais tarde.

Dados de desempenho

O Nagios oferece a possibilidade de receber dos *plugins* dados de desempenho, em adição aos comuns dados de estado, bem como fornecer de seguida esses dados a aplicações externas para processamento.

São distinguidos dois tipos de dados de desempenho. Os dados de desempenho da verificação e os dados de desempenho específicos do *plugin*.
Os dados de desempenho da verificação são constituídos por informação relativa à verificação em si, como, por exemplo, atraso da verificação em relação à marcação estipulada, ou quanto tempo demorou a verificação a ser efectuada. Este tipo de informação está disponível para qualquer verificação efectuada, não sendo necessário definir qualquer tipo de código para a determinar. Basta ir buscá-la às macros correspondentes.
Os dados de desempenho específicos do *plugin*, são dados, normalmente, intimamente relacionados com o serviço monitorizado. Podem conter valores de temperatura, carga de CPU, ou qualquer outro tipo de medição feita aquando da execução do *plugin*. É uma informação disponibilizada opcionalmente, logo, não é suportada por todos os *plugins*.

Pode haver necessidade de efectuar algum tipo de processamento automático sobre os dados de desempenho. O Nagios prevê dois métodos para efectuar esta tarefa. Um método por defeito, em que é executado um comando para processamento especificado num ficheiro, ou um método "*file-based*", em que os dados são despejados, tal e qual como foram recebidos, para um ficheiro.

Suporte para bases de dados

O Nagios possibilita o armazenamento de diversos tipos de informação em bases de dados estruturadas. Actualmente só estão disponíveis bases de dados em MySQL e PostGreSQL, porém é de prever que no futuro muitas mais sejam suportadas.

Capítulo 11 – Crítica Construtiva ao Nagios

O Nagios não é um *software* apropriado para quem quer só correr um programa instalador e ficar logo com as coisas a correr. Para além disto o caso é ainda pior quando falamos de gestão de redes no real sentido da palavra, ou seja, a realização desta última parte é bem mais complexa e trabalhosa do que efectuar apenas a monitorização de redes. Pode-se dizer que fazer gestão de redes com Nagios implica ao novo utilizador um esforço grande para a compreensão e aplicação de todos os novos conceitos e técnicas de funcionamento do Nagios. Contudo não se deve ver isto como um ponto negativo, mas sim como um dos requisitos para ser administrador de Nagios.

O Nagios não foi desenhado para substituir na íntegra uma aplicação de gestão SNMP, como o HP OpenView ou o OpenNMS. No entanto é possível definir as coisas de maneira a que *SNMP traps* recebidas por um *host* na nossa rede gerem alertas no Nagios. A possibilidade de se poder integrar Nagios com outras ferramentas (Portsentry, UCD-SNMP, TCP wrapper, etc.) é uma grande vantagem em relação à utilidade da ferramenta. Esta parece ser a grande valia do Nagios em relação a outros produtos.

Capítulo 12 – Últimos Desenvolvimentos

Actualmente existe uma versão alpha 2.0 e uma versão 3.0 em fase embrionária. Algumas das melhorias a verificar numa versão utilizável em ambientes de produção são:

suporte para verificação passiva de *hosts*, tal como existe actualmente com verificação passiva de serviços (2.0);
adição de *servicegroups* aos tipos de objectos definidos – estes vão ser opcionais e vão permitir criar conjuntos de serviços com o propósito de mostrar o estado de processos de negócio, etc. (2.0);
funcionalidades de monitorização adaptativa – habilidade de alterar como a verificação de serviços e *hosts* é realizada em tempo de execução – isto vai permitir escrever *event handlers* ou scripts externos que modificam parâmetros de monitorização de *hosts* e serviços (2.0);
despoletar de tempos inactivos, o que é bastante útil se considerarmos que queremos agendar o tempo de inactividade para um *host* e despoletar um agendamento de inactividade para todos os seus *hosts* "filhos" (2.0);
saída de informação em tempo real (2.0);
verificação em paralelo e inteligente de *hosts* (3.0);
interface em PHP a substituir os CGIs (3.0).

Capítulo 13 – Conclusão

Com este trabalho podemos concluir que o Nagios pode ser uma ferramenta de monitorização de *hosts* muito útil. O GUI profissional que o Nagios oferece rivaliza com muitos de ferramentas comerciais. A habilidade do Nagios poder fazer monitorização remota sem instalação de novo *software* é uma mais valia. Para além disto este *software* é gratuito e livre, o que o torna num candidato ideal para qualquer organização que queira instalar um sistema de gestão de redes. Como isto tudo justifica-se o grande número de organizações a usar Nagios, nomeadamente empresas, universidades e organismos governamentais.

Bibliografia

"Nagios version 1.0 documentation".
Ethan Galstad & Daniel Koffler, 2002

"Apache HTTP Server Version 1.3 Documentation".
The Apache Software Foundation

"Documentação dos Plugins 1.3.0 do Nagios".
Diversos Autores

"GNU General Public License"
Free Software Foundation 1989, 1991

"Network Monitoring with Nagios"
Syed Ali, October 2003

"Nagios® Configurations"
squareBOX technologies, 2003

"Installing Nagios"& "Configuring Monitoring"
Oktay Altunergil in O'Reilly ONLamp.com , 2002

"Evalluaciión de Nagiios para Liinux"
José A. Zarandieta & Manuel Domínguez, 2003

"PROCEDURE FOR THE INSTALLATION OF THE NAGIOS NETWORK MONITORING PROGRAM"
Andrew Kaplan

ANEXO A – Ficheiros de Configuração

```
##############################################################################
#
# NAGIOS.CFG - Ficheiro de Configuração Principal
#
##############################################################################


# LOG FILE
# This is the main log file where service and host events are logged
# for historical purposes.  This should be the first option specified
# in the config file!!!

log_file=/var/spool/nagios/nagios.log



# OBJECT CONFIGURATION FILE(S)
# This is the configuration file in which you define hosts, host
# groups, contacts, contact groups, services, etc.  I guess it would
# be better called an object definition file, but for historical
# reasons it isn't.  You can split object definitions into several
# different config files by using multiple cfg_file statements here.
# Nagios will read and process all the config files you define.
# This can be very useful if you want to keep command definitions
# separate from host and contact definitions...

# Plugin commands (service and host check commands)
# Arguments are likely to change between different releases of the
# plugins, so you should use the same config file provided with the
# plugin release rather than the one provided with Nagios.
cfg_file=/usr/local/etc/nagios/checkcommands.cfg

# Misc commands (notification and event handler commands, etc)
cfg_file=/usr/local/etc/nagios/misccommands.cfg

# You can split other types of object definitions across several
# config files if you wish (as done here), or keep them all in a
# single config file.

cfg_file=/usr/local/etc/nagios/contactgroups.cfg
cfg_file=/usr/local/etc/nagios/contacts.cfg
cfg_file=/usr/local/etc/nagios/dependencies.cfg
cfg_file=/usr/local/etc/nagios/escalations.cfg
cfg_file=/usr/local/etc/nagios/hostgroups.cfg
cfg_file=/usr/local/etc/nagios/hosts.cfg
cfg_file=/usr/local/etc/nagios/services.cfg
cfg_file=/usr/local/etc/nagios/timeperiods.cfg


# RESOURCE FILE
# This is an optional resource file that contains $USERx$ macro
# definitions. Multiple resource files can be specified by using
# multiple resource_file definitions.  The CGIs will not attempt to
# read the contents of resource files, so information that is
# considered to be sensitive (usernames, passwords, etc) can be
# defined as macros in this file and restrictive permissions (600)
# can be placed on this file.

resource_file=/usr/local/etc/nagios/resource.cfg



# STATUS FILE
# This is where the current status of all monitored services and
# hosts is stored.  Its contents are read and processed by the CGIs.
# The contentsof the status file are deleted every time Nagios
#  restarts.

status_file=/var/spool/nagios/status.log
```

```
# NAGIOS USER
# This determines the effective user that Nagios should run as.
# You can either supply a username or a UID.

nagios_user=nagios



# NAGIOS GROUP
# This determines the effective group that Nagios should run as.
# You can either supply a group name or a GID.

nagios_group=nagios



# EXTERNAL COMMAND OPTION
# This option allows you to specify whether or not Nagios should check
# for external commands (in the command file defined below).  By default
# Nagios will *not* check for external commands, just to be on the
# cautious side.  If you want to be able to use the CGI command interface
# you will have to enable this.  Setting this value to 0 disables command
# checking (the default), other values enable it.

check_external_commands=1



# EXTERNAL COMMAND CHECK INTERVAL
# This is the interval at which Nagios should check for external commands.
# This value works of the interval_length you specify later.  If you leave
# that at its default value of 60 (seconds), a value of 1 here will cause
# Nagios to check for external commands every minute.  If you specify a
# number followed by an "s" (i.e. 15s), this will be interpreted to mean
# actual seconds rather than a multiple of the interval_length variable.
# Note: In addition to reading the external command file at regularly
# scheduled intervals, Nagios will also check for external commands after
# event handlers are executed.
# NOTE: Setting this value to -1 causes Nagios to check the external
# command file as often as possible.

#command_check_interval=1
#command_check_interval=15s
command_check_interval=-1



# EXTERNAL COMMAND FILE
# This is the file that Nagios checks for external command requests.
# It is also where the command CGI will write commands that are submitted
# by users, so it must be writeable by the user that the web server
# is running as (usually 'nobody').  Permissions should be set at the
# directory level instead of on the file, as the file is deleted every
# time its contents are processed.

command_file=/var/spool/nagios/rw/nagios.cmd



# COMMENT FILE
# This is the file that Nagios will use for storing host and service
# comments.

comment_file=/var/spool/nagios/comment.log



# DOWNTIME FILE
# This is the file that Nagios will use for storing host and service
# downtime data.
```

```
downtime_file=/var/spool/nagios/downtime.log



# LOCK FILE
# This is the lockfile that Nagios will use to store its PID number
# in when it is running in daemon mode.

lock_file=/var/spool/nagios/nagios.lock



# TEMP FILE
# This is a temporary file that is used as scratch space when Nagios
# updates the status log, cleans the comment file, etc.  This file
# is created, used, and deleted throughout the time that Nagios is
# running.

temp_file=/var/spool/nagios/nagios.tmp



# LOG ROTATION METHOD
# This is the log rotation method that Nagios should use to rotate
# the main log file. Values are as follows..
#        n         = None - don't rotate the log
#        h         = Hourly rotation (top of the hour)
#        d         = Daily rotation (midnight every day)
#        w         = Weekly rotation (midnight on Saturday evening)
#        m         = Monthly rotation (midnight last day of month)

log_rotation_method=d



# LOG ARCHIVE PATH
# This is the directory where archived (rotated) log files should be
# placed (assuming you've chosen to do log rotation).

log_archive_path=/var/spool/nagios/archives



# LOGGING OPTIONS
# If you want messages logged to the syslog facility, as well as the
# NetAlarm log file set this option to 1.  If not, set it to 0.

use_syslog=1



# NOTIFICATION LOGGING OPTION
# If you don't want notifications to be logged, set this value to 0.
# If notifications should be logged, set the value to 1.

log_notifications=1



# SERVICE RETRY LOGGING OPTION
# If you don't want service check retries to be logged, set this value
# to 0.  If retries should be logged, set the value to 1.

log_service_retries=1



# HOST RETRY LOGGING OPTION
# If you don't want host check retries to be logged, set this value to
# 0.  If retries should be logged, set the value to 1.
```

```
log_host_retries=1



# EVENT HANDLER LOGGING OPTION
# If you don't want host and service event handlers to be logged, set
# this value to 0.  If event handlers should be logged, set the value
# to 1.

log_event_handlers=1



# INITIAL STATES LOGGING OPTION
# If you want Nagios to log all initial host and service states to
# the main log file (the first time the service or host is checked)
# you can enable this option by setting this value to 1.  If you
# are not using an external application that does long term state
# statistics reporting, you do not need to enable this option.  In
# this case, set the value to 0.

log_initial_states=0



# EXTERNAL COMMANDS LOGGING OPTION
# If you don't want Nagios to log external commands, set this value
# to 0.  If external commands should be logged, set this value to 1.
# Note: This option does not include logging of passive service
# checks - see the option below for controlling whether or not
# passive checks are logged.

log_external_commands=1



# PASSIVE SERVICE CHECKS LOGGING OPTION
# If you don't want Nagios to log passive service checks, set this
# value to 0.  If passive service checks should be logged, set this
# value to 1.

log_passive_service_checks=1



# GLOBAL HOST AND SERVICE EVENT HANDLERS
# These options allow you to specify a host and service event handler
# command that is to be run for every host or service state change.
# The global event handler is executed immediately prior to the event
# handler that you have optionally specified in each host or
# service definition. The command argument is the short name of a
# command definition that you define in your host configuration file.
# Read the HTML docs for more information.

#global_host_event_handler=somecommand
#global_service_event_handler=somecommand



# INTER-CHECK DELAY METHOD
# This is the method that Nagios should use when initially
# "spreading out" service checks when it starts monitoring.  The
# default is to use smart delay calculation, which will try to
# space all service checks out evenly to minimize CPU load.
# Using the dumb setting will cause all checks to be scheduled
# at the same time (with no delay between them)!  This is not a
# good thing for production, but is useful when testing the
# parallelization functionality.
#        n        = None - don't use any delay between checks
#        d        = Use a "dumb" delay of 1 second between checks
```

```
#         s         = Use "smart" inter-check delay calculation
#       x.xx    = Use an inter-check delay of x.xx seconds

inter_check_delay_method=s



# SERVICE CHECK INTERLEAVE FACTOR
# This variable determines how service checks are interleaved.
# Interleaving the service checks allows for a more even
# distribution of service checks and reduced load on remote
# hosts.  Setting this value to 1 is equivalent to how versions
# of Nagios previous to 0.0.5 did service checks.  Set this
# value to s (smart) for automatic calculation of the interleave
# factor unless you have a specific reason to change it.
#       s       = Use "smart" interleave factor calculation
#       x       = Use an interleave factor of x, where x is a
#                 number greater than or equal to 1.

service_interleave_factor=s



# MAXIMUM CONCURRENT SERVICE CHECKS
# This option allows you to specify the maximum number of
# service checks that can be run in parallel at any given time.
# Specifying a value of 1 for this variable essentially prevents
# any service checks from being parallelized.  A value of 0
# will not restrict the number of concurrent checks that are
# being executed.

max_concurrent_checks=0



# SERVICE CHECK REAPER FREQUENCY
# This is the frequency (in seconds!) that Nagios will process
# the results of services that have been checked.

service_reaper_frequency=10



# SLEEP TIME
# This is the number of seconds to sleep between checking for system
# events and service checks that need to be run.  I would recommend
# *not* changing this from its default value of 1 second.

sleep_time=1



# TIMEOUT VALUES
# These options control how much time Nagios will allow various
# types of commands to execute before killing them off.  Options
# are available for controlling maximum time allotted for
# service checks, host checks, event handlers, notifications, the
# ocsp command, and performance data commands.  All values are in
# seconds.

service_check_timeout=60
host_check_timeout=30
event_handler_timeout=30
notification_timeout=30
ocsp_timeout=5
perfdata_timeout=5



# RETAIN STATE INFORMATION
# This setting determines whether or not Nagios will save state
```

```
# information for services and hosts before it shuts down.  Upon
# startup Nagios will reload all saved service and host state
# information before starting to monitor.  This is useful for
# maintaining long-term data on state statistics, etc, but will
# slow Nagios down a bit when it (re)starts.  Since its only
# a one-time penalty, I think its well worth the additional
# startup delay.

retain_state_information=1



# STATE RETENTION FILE
# This is the file that Nagios should use to store host and
# service state information before it shuts down.  The state
# information in this file is also read immediately prior to
# starting to monitor the network when Nagios is restarted.
# This file is used only if the preserve_state_information
# variable is set to 1.

state_retention_file=/var/spool/nagios/status.sav



# RETENTION DATA UPDATE INTERVAL
# This setting determines how often (in minutes) that Nagios
# will automatically save retention data during normal operation.
# If you set this value to 0, Nagios will not save retention
# data at regular interval, but it will still save retention
# data before shutting down or restarting.  If you have disabled
# state retention, this option has no effect.

retention_update_interval=60



# USE RETAINED PROGRAM STATE
# This setting determines whether or not Nagios will set
# program status variables based on the values saved in the
# retention file.  If you want to use retained program status
# information, set this value to 1.  If not, set this value
# to 0.

use_retained_program_state=0



# INTERVAL LENGTH
# This is the seconds per unit interval as used in the
# host/contact/service configuration files.  Setting this to 60 means
# that each interval is one minute long (60 seconds).  Other settings
# have not been tested much, so your mileage is likely to vary...

interval_length=60



# AGRESSIVE HOST CHECKING OPTION
# If you don't want to turn on agressive host checking features, set
# this value to 0 (the default).  Otherwise set this value to 1 to
# enable the agressive check option.  Read the docs for more info
# on what agressive host check is or check out the source code in
# base/checks.c

use_agressive_host_checking=0



# SERVICE CHECK EXECUTION OPTION
# This determines whether or not Nagios will actively execute
# service checks when it initially starts.  If this option is
```

```
# disabled, checks are not actively made, but Nagios can still
# receive and process passive check results that come in.  Unless
# you're implementing redundant hosts or have a special need for
# disabling the execution of service checks, leave this enabled!
# Values: 1 = enable checks, 0 = disable checks

execute_service_checks=1



# PASSIVE CHECK ACCEPTANCE OPTION
# This determines whether or not Nagios will accept passive
# service checks results when it initially (re)starts.
# Values: 1 = accept passive checks, 0 = reject passive checks

accept_passive_service_checks=1



# NOTIFICATIONS OPTION
# This determines whether or not Nagios will sent out any host or
# service notifications when it is initially (re)started.
# Values: 1 = enable notifications, 0 = disable notifications

enable_notifications=1



# EVENT HANDLER USE OPTION
# This determines whether or not Nagios will run any host or
# service event handlers when it is initially (re)started.  Unless
# you're implementing redundant hosts, leave this option enabled.
# Values: 1 = enable event handlers, 0 = disable event handlers

enable_event_handlers=1



# PROCESS PERFORMANCE DATA OPTION
# This determines whether or not Nagios will process performance
# data returned from service and host checks.  If this option is
# enabled, host performance data will be processed using the
# host_perfdata_command (defined below) and service performance
# data will be processed using the service_perfdata_command (also
# defined below).  Read the HTML docs for more information on
# performance data.
# Values: 1 = process performance data, 0 = do not process performance data

process_performance_data=0



# HOST AND SERVICE PERFORMANCE DATA PROCESSING COMMANDS
# These commands are run after every host and service check is
# performed.  These commands are executed only if the
# enable_performance_data option (above) is set to 1.  The command
# argument is the short name of a command definition that you
# define in your host configuration file.  Read the HTML docs for
# more information on performance data.

#host_perfdata_command=process-host-perfdata
#service_perfdata_command=process-service-perfdata



# OBSESS OVER SERVICE CHECKS OPTION
# This determines whether or not Nagios will obsess over service
# checks and run the ocsp_command defined below.  Unless you're
# planning on implementing distributed monitoring, do not enable
# this option.  Read the HTML docs for more information on
# implementing distributed monitoring.
```

```
# Values: 1 = obsess over services, 0 = do not obsess (default)

obsess_over_services=0



# OBSESSIVE COMPULSIVE SERVICE PROCESSOR COMMAND
# This is the command that is run for every service check that is
# processed by Nagios.  This command is executed only if the
# obsess_over_service option (above) is set to 1.  The command
# argument is the short name of a command definition that you
# define in your host configuration file. Read the HTML docs for
# more information on implementing distributed monitoring.

#ocsp_command=somecommand



# ORPHANED SERVICE CHECK OPTION
# This determines whether or not Nagios will periodically
# check for orphaned services.  Since service checks are not
# rescheduled until the results of their previous execution
# instance are processed, there exists a possibility that some
# checks may never get rescheduled.  This seems to be a rare
# problem and should not happen under normal circumstances.
# If you have problems with service checks never getting
# rescheduled, you might want to try enabling this option.
# Values: 1 = enable checks, 0 = disable checks

check_for_orphaned_services=0



# SERVICE FRESHNESS CHECK OPTION
# This option determines whether or not Nagios will periodically
# check the "freshness" of service results.  Enabling this option
# is useful for ensuring passive checks are received in a timely
# manner.
# Values: 1 = enabled freshness checking, 0 = disable freshness checking

check_service_freshness=1



# FRESHNESS CHECK INTERVAL
# This setting determines how often (in seconds) Nagios will
# check the "freshness" of service check results.  If you have
# disabled service freshness checking, this option has no effect.

freshness_check_interval=60



# AGGREGATED STATUS UPDATES
# This option determines whether or not Nagios will
# aggregate updates of host, service, and program status
# data.  Normally, status data is updated immediately when
# a change occurs.  This can result in high CPU loads if
# you are monitoring a lot of services.  If you want Nagios
# to only refresh status data every few seconds, disable
# this option.
# Values: 1 = enable aggregate updates, 0 = disable aggregate updates

aggregate_status_updates=1



# AGGREGATED STATUS UPDATE INTERVAL
# Combined with the aggregate_status_updates option,
# this option determines the frequency (in seconds!) that
# Nagios will periodically dump program, host, and
```

```
# service status data.  If you are not using aggregated
# status data updates, this option has no effect.

status_update_interval=15



# FLAP DETECTION OPTION
# This option determines whether or not Nagios will try
# and detect hosts and services that are "flapping".
# Flapping occurs when a host or service changes between
# states too frequently.  When Nagios detects that a
# host or service is flapping, it will temporarily supress
# notifications for that host/service until it stops
# flapping.  Flap detection is very experimental, so read
# the HTML documentation before enabling this feature!
# Values: 1 = enable flap detection
#         0 = disable flap detection (default)

enable_flap_detection=0



# FLAP DETECTION THRESHOLDS FOR HOSTS AND SERVICES
# Read the HTML documentation on flap detection for
# an explanation of what this option does.  This option
# has no effect if flap detection is disabled.

low_service_flap_threshold=5.0
high_service_flap_threshold=20.0
low_host_flap_threshold=5.0
high_host_flap_threshold=20.0



# DATE FORMAT OPTION
# This option determines how short dates are displayed. Valid options
# include:
#         us                  (MM-DD-YYYY HH:MM:SS)
#         euro      (DD-MM-YYYY HH:MM:SS)
#         iso8601             (YYYY-MM-DD HH:MM:SS)
#         strict-iso8601      (YYYY-MM-DDTHH:MM:SS)
#

date_format=euro



# ILLEGAL OBJECT NAME CHARACTERS
# This options allows you to specify illegal characters that cannot
# be used in host names, service descriptions, or names of other
# object types.

illegal_object_name_chars=`~!$%^&*|'"<>?,()=



# ILLEGAL MACRO OUTPUT CHARACTERS
# This options allows you to specify illegal characters that are
# stripped from macros before being used in notifications, event
# handlers, etc.  This DOES NOT affect macros used in service or
# host check commands.
# The following macros are stripped of the characters you specify:
#         $OUTPUT$, $PERFDATA$

illegal_macro_output_chars=`~$&|'"<>



# ADMINISTRATOR EMAIL ADDRESS
# The email address of the administrator of *this* machine (the one
```

```
# doing the monitoring).  Nagios never uses this value itself, but
# you can access this value by using the $ADMINEMAIL$ macro in your
# notification commands.

admin_email=nagios



# ADMINISTRATOR PAGER NUMBER/ADDRESS
# The pager number/address for the administrator of *this* machine.
# Nagios never uses this value itself, but you can access this
# value by using the $ADMINPAGER$ macro in your notification
# commands.

admin_pager=pagenagios



# EOF (End of file)
```

```
##############################################################
#
# CGI.CFG - Ficheiro de Configuração dos CGIs
#
##############################################################


# MAIN CONFIGURATION FILE
# This tells the CGIs where to find your main configuration file.
# The CGIs will read the main and host config files for any other
# data they might need.

main_config_file=/usr/local/etc/nagios/nagios.cfg



# PHYSICAL HTML PATH
# This is the path where the HTML files for Nagios reside.  This
# value is used to locate the logo images needed by the statusmap
# and statuswrl CGIs.

physical_html_path=/usr/local/share/nagios



# URL HTML PATH
# This is the path portion of the URL that corresponds to the
# physical location of the Nagios HTML files (as defined above).
# This value is used by the CGIs to locate the online documentation
# and graphics.  If you access the Nagios pages with an URL like
# http://www.myhost.com/nagios, this value should be '/nagios'
# (without the quotes).

url_html_path=/nagios



# CONTEXT-SENSITIVE HELP
# This option determines whether or not a context-sensitive
# help icon will be displayed for most of the CGIs.
# Values: 0 = disables context-sensitive help
#         1 = enables context-sensitive help

show_context_help=0



# NAGIOS PROCESS CHECK COMMAND
# This is the full path and filename of the program used to check
# the status of the Nagios process.  It is used only by the CGIs
# and is completely optional.  However, if you don't use it, you'll
# see warning messages in the CGIs about the Nagios process
# not running and you won't be able to execute any commands from
# the web interface.  The program should follow the same rules
# as plugins; the return codes are the same as for the plugins,
# it should have timeout protection, it should output something
# to STDIO, etc.
#
# Note: If you are using the check_nagios plugin here, the first
# argument should be the physical path to the status log, the
# second argument is the number of minutes that the status log
# contents should be "fresher" than, and the third argument is the
# string that should be matched from the output of the 'ps'
# command in order to locate the running Nagios process.  That
# process string is going to vary depending on how you start
# Nagios.  Run the 'ps' command manually to see what the command
# line entry for the Nagios process looks like.

#nagios_check_command=/usr/local/libexec/nagios/check_nagios  /var/spool/nagios/status.log
5 '/usr/local/bin/nagios'
```

```
# AUTHENTICATION USAGE
# This option controls whether or not the CGIs will use any
# authentication when displaying host and service information, as
# well as committing commands to Nagios for processing.
#
# Read the HTML documentation to learn how the authorization works!
#
# NOTE: It is a really *bad* idea to disable authorization, unless
# you plan on removing the command CGI (cmd.cgi)!  Failure to do
# so will leave you wide open to kiddies messing with Nagios and
# possibly hitting you with a denial of service attack by filling up
# your drive by continuously writing to your command file!
#
# Setting this value to 0 will cause the CGIs to *not* use
# authentication (bad idea), while any other value will make them
# use the authentication functions (the default).

use_authentication=1



# DEFAULT USER
# Setting this variable will define a default user name that can
# access pages without authentication.  This allows people within a
# secure domain (i.e., behind a firewall) to see the current status
# without authenticating.  You may want to use this to avoid basic
# authentication if you are not using a sercure server since basic
# authentication transmits passwords in the clear.
#
# Important:  Do not define a default username unless you are
# running a secure web server and are sure that everyone who has
# access to the CGIs has been authenticated in some manner!  If you
# define this variable, anyone who has not authenticated to the web
# server will inherit all rights you assign to this user!

#default_user_name=guest



# SYSTEM/PROCESS INFORMATION ACCESS
# This option is a comma-delimited list of all usernames that
# have access to viewing the Nagios process information as
# provided by the Extended Information CGI (extinfo.cgi).  By
# default, *no one* has access to this unless you choose to
# not use authorization.  You may use an asterisk (*) to
# authorize any user who has authenticated to the web server.

authorized_for_system_information=davidmar,guru,vmac



# CONFIGURATION INFORMATION ACCESS
# This option is a comma-delimited list of all usernames that
# can view ALL configuration information (hosts, commands, etc).
# By default, users can only view configuration information
# for the hosts and services they are contacts for. You may use
# an asterisk (*) to authorize any user who has authenticated
# to the web server.

authorized_for_configuration_information=davidmar,guru



# SYSTEM/PROCESS COMMAND ACCESS
# This option is a comma-delimited list of all usernames that
# can issue shutdown and restart commands to Nagios via the
# command CGI (cmd.cgi).  Users in this list can also change
# the program mode to active or standby. By default, *no one*
# has access to this unless you choose to not use authorization.
```

```
# You may use an asterisk (*) to authorize any user who has
# authenticated to the web server.

authorized_for_system_commands=guru



# GLOBAL HOST/SERVICE VIEW ACCESS
# These two options are comma-delimited lists of all usernames that
# can view information for all hosts and services that are being
# monitored.  By default, users can only view information
# for hosts or services that they are contacts for (unless you
# you choose to not use authorization). You may use an asterisk (*)
# to authorize any user who has authenticated to the web server.


authorized_for_all_services=guru
authorized_for_all_hosts=guru



# GLOBAL HOST/SERVICE COMMAND ACCESS
# These two options are comma-delimited lists of all usernames that
# can issue host or service related commands via the command
# CGI (cmd.cgi) for all hosts and services that are being monitored.
# By default, users can only issue commands for hosts or services
# that they are contacts for (unless you you choose to not use
# authorization).  You may use an asterisk (*) to authorize any
# user who has authenticated to the web server.

authorized_for_all_service_commands=guru
authorized_for_all_host_commands=guru



# EXTENDED HOST INFORMATION
# This is all entirely optional.  If you don't enter any extended
# information, nothing bad will happen - I promise...  Its basically
# just used to have pretty icons and such associated with your hosts.
# This is especially nice when you're using the statusmap and
# statuswrl CGIs.  You can also specify an URL that links to a document
# containing more information about the host (location details, contact
# information, etc).
#
# hostextinfo[<host_name>]=<notes_url>;<icon_image>;<vrml_image>;<gd2_image>;\
#                               <image_alt>;<x_2d>,<y_2d>;<x_3d>,<y_3d>,<z_3d>;

#hostextinfo[surecom]=/usr/local/share/nagios/ext/surecom.html;/usr/local/share/nagios/ext
/surecom.png
#hostextinfo[oriental]=/usr/local/share/nagios/ext/oriental.html;/usr/local/share/nagios/e
xt/oriental.jpg
#hostextinfo[241s1]=/usr/local/share/nagios/ext/241s1.html;/usr/local/share/nagios/ext/241
s1.png
xedtemplate_config_file=/usr/local/etc/nagios/hostextinfo.cfg


#
# <notes_url>        = Optional URL that points to a document of
#                      some type containing information on the host.
#                      The information (and the document type) can
#                      be anything you want.  Examples include details
#                      on the physical location of the server, info
#                      on how to contact the admins in case of an
#                      emergency, etc.  Relative URLs start in the
#                      same path that is used to access the CGIs.
#                      The link that is created for the host's notes
#                      notes is found in the extinfo CGI.
#                      Note: You may use the $HOSTNAME$ and
#                      $HOSTADDRESS$ macros in this URL.
# <icon_image>       = A GIF, PNG, or JPEG image to associate with
#                      the host.  This is used in the status and
```

```
#                       extinfo CGIs.
# <vrml_image>        = An image to use in the statuswrl CGI in the
#                       VRML generation.  Transparent images don't
#                       work so great..
# <gd2_image>         = An image used by the statusmap CGI to
#                       represent the host.  This can be a GIF, PNG,
#                       JPEG, or GD2 image.  GD2 format is recommended,
#                       as it produces the load CPU load.
#                       utility supplied with Boutell's gd library.
# <image_alt>         = ALT tag used with images in various CGIs
# <x_2d>,<y_2d>       = X and Y coordinates used when drawing the
#                       host in the statusmap CGI.  (0,0) is located
#                       in the upper left corner of the screen and is
#                       considered to be the origin.  The coordinates
#                       you supply here are used as the coords of the
#                       upper left hand corner of host icon.  Both
#                       numbers should be positive integers.
# <x_3d>,<y_3d>,<z_3d> = X, Y, and Z coordinates used when drawing
#                       the host in the statuswrl (VRML) CGI.  All
#                       numbers can be positive or negative (anywhere
#                       in 3-D space).  The coordinates are used to
#                       determine the center of the host "cube" that
#                       is drawn. Host "cubes" are drawn with a
#                       height, width, and depth of 0.5 (meters).
#
# Note: All images must be placed in the /logos subdirectory under
# the HTML images path (i.e. /usr/local/nagios/share/images/logos/).
# This path is automatically determined by appending "/images/logos"
# to the path specified by the 'physical_html_path' directive.

#hostextinfo[es-eds]=/serverinfo/es-
eds.html;novell40.gif;novell40.jpg;novell40.gd2;IntranetWare 4.11;100,50;3.5,0.0,-1.5;
#hostextinfo[rosie]=/serverinfo/rosie.html;win40.gif;win40.jpg;win40.gd2;NT Server 4.0;;;



# EXTENDED SERVICE INFORMATION
# This is all entirely optional.  If you don't enter any extended
# information, nothing bad will happen - I promise...  Its basically
# just used to have pretty icons and such associated with your services.
# You can also specify an URL that links to a document containing more
# information about the service (location details, contact information,
# etc).
#
# serviceextinfo[<host_name>;<svc_description>]=<notes_url>;<icon_image>;<image_alt>
#
# <notes_url>         = Optional URL that points to a document of
#                       some type containing information on the service.
#                       The information (and the document type) can
#                       be anything you want.  Examples include details
#                       on the physical location of the server, info
#                       on how to contact the admins in case of an
#                       emergency, etc.  Relative URLs start in the
#                       same path that is used to access the CGIs.
#                       The link that is created for the service's
#                       notes URL is found in the extinfo CGI.
#                       Note: You may use the $HOSTNAME$, $HOSTADDRESS$,
#                       and $SERVICEDESC$ macros in this URL.
# <icon_image>        = A GIF, PNG, or JPEG image to associate with
#                       the service.  This is used in the status and
#                       extinfo CGIs.
# <image_alt>         = ALT tag used with image
#
# Note: All images must be placed in the /logos subdirectory under
# the HTML images path (i.e. /usr/local/nagios/share/images/logos/).
# This path is automatically determined by appending "/images/logos"
# to the path specified by the 'physical_html_path' directive.

#serviceextinfo[es-eds;PING]=http://www.somewhere.com?traceroute&to=$HOSTADDRESS$;;PING
rate
#serviceextinfo[rosie;Security Alerts]=;security.gif;Security alerts
```

```
# STATUSMAP BACKGROUND IMAGE
# This option allows you to specify an image to be used as a
# background in the statusmap CGI.  It is assumed that the image
# resides in the HTML images path (i.e. /usr/local/nagios/share/images).
# This path is automatically determined by appending "/images"
# to the path specified by the 'physical_html_path' directive.
# Note:  The image file may be in GIF, PNG, JPEG, or GD2 format.
# However, I recommend that you convert your image to GD2 format
# (uncompressed), as this will cause less CPU load when the CGI
# generates the image.

#statusmap_background_image=smbackground.gd2



# DEFAULT STATUSMAP LAYOUT METHOD
# This option allows you to specify the default layout method
# the statusmap CGI should use for drawing hosts.  If you do
# not use this option, the default is to use user-defined
# coordinates.  Valid options are as follows:
#         0 = User-defined coordinates
#         1 = Depth layers
#       2 = Collapsed tree
#       3 = Balanced tree
#       4 = Circular
#       5 = Circular (Marked Up)

default_statusmap_layout=5



# DEFAULT STATUSWRL LAYOUT METHOD
# This option allows you to specify the default layout method
# the statuswrl (VRML) CGI should use for drawing hosts.  If you
# do not use this option, the default is to use user-defined
# coordinates.  Valid options are as follows:
#         0 = User-defined coordinates
#       2 = Collapsed tree
#       3 = Balanced tree
#       4 = Circular

default_statuswrl_layout=4



# STATUSWRL INCLUDE
# This option allows you to include your own objects in the
# generated VRML world.  It is assumed that the file
# resides in the HTML path (i.e. /usr/local/nagios/share).

#statuswrl_include=myworld.wrl



# PING SYNTAX
# This option determines what syntax should be used when
# attempting to ping a host from the WAP interface (using
# the statuswml CGI.  You must include the full path to
# the ping binary, along with all required options.  The
# $HOSTADDRESS$ macro is substituted with the address of
# the host before the command is executed.

ping_syntax=/bin/ping -n -U -c 5 $HOSTADDRESS$



# REFRESH RATE
# This option allows you to specify the refresh rate in seconds
```

```
# of various CGIs (status, statusmap, extinfo, and outages).

refresh_rate=90



# SOUND OPTIONS
# These options allow you to specify an optional audio file
# that should be played in your browser window when there are
# problems on the network.  The audio files are used only in
# the status CGI.  Only the sound for the most critical problem
# will be played.  Order of importance (higher to lower) is as
# follows: unreachable hosts, down hosts, critical services,
# warning services, and unknown services. If there are no
# visible problems, the sound file optionally specified by
# 'normal_sound' variable will be played.
#
#
# <varname>=<sound_file>
#
# Note: All audio files must be placed in the /media subdirectory
# under the HTML path (i.e. /usr/local/nagios/share/media/).

#host_unreachable_sound=hostdown.wav
#host_down_sound=hostdown.wav
#service_critical_sound=critical.wav
#service_warning_sound=warning.wav
#service_unknown_sound=warning.wav
#normal_sound=noproblem.wav



# DG EXTENDED DATA
# Note: These config directives are only used if you compiled
# in database support for extended data!
# The user you specify here only needs SELECT privileges on the
# 'hostextinfo' table in the database.

#xeddb_host=somehost
#xeddb_port=someport
#xeddb_database=somedatabase
#xeddb_username=someuser
#xeddb_password=somepassword



# DB STATUS DATA (Read-Only For CGIs)
# Note: These config directives are only used if you compiled
# in database support for status data!
# The user you specify here only needs SELECT privileges on the
# 'programstatus', 'hoststatus', and 'servicestatus' tables
# in the database, as these values are only used by the CGIs.
# The core program will read the directives you specify in
# in a resource file.

#xsddb_host=somehost
#xsddb_port=someport
#xsddb_database=somedatabase
#xsddb_username=someuser
#xsddb_password=somepassword



# DB COMMENT DATA (Read-Only For CGIs)
# Note: These config directives are only used if you compiled
# in database support for comment data!
# The user you specify here only needs SELECT privileges on the
# 'hostcomments', and 'servicecomments' tables in the database,
# as these values are only used by the CGIs.  The core program
# will read the directives you specify in a resource file.
```

```
#xcddb_host=somehost
#xcddb_port=someport
#xcddb_database=somedatabase
#xcddb_username=someuser
#xcddb_password=somepassword



# DB DOWNTIME DATA (Read-Only For CGIs)
# Note: These config directives are only used if you compiled
# in database support for downtime data!
# The user you specify here only needs SELECT privileges on the
# 'hostdowntime', and 'servicedowntime' tables in the database,
# as these values are only used by the CGIs.  The core program
# will read the directives you specify in a resource file.

#xdddb_host=somehost
#xdddb_port=someport
#xdddb_database=somedatabase
#xdddb_username=someuser
#xdddb_password=somepassword
```

```
##############################################################################
#
# CHECKCOMMANDS.CFG
#
##############################################################################


##############################################################################
# COMMAND DEFINITIONS
#
# SYNTAX:
#
#         define command{
#                template      <templatename>
#                   name          <objectname>
#                command_name  <commandname>
#                command_line  <commandline>
#                }
#
# WHERE:
#
# <templatename> = object name of another command definition that should be
#                  used as a template for this definition (optional)
# <objectname>   = object name of command definition, referenced by other
#                  command definitions that use it as a template (optional)
# <commandname>  = name of the command, as recognized/used by Nagios
# <commandline>  = command line
#
##############################################################################


# 'check_tcp' command definition
define command{
          command_name       check_tcp
          command_line       $USER1$/check_tcp -H $HOSTADDRESS$ -p $ARG1$
          }


# 'check_udp' command definition
define command{
          command_name       check_udp
          command_line       $USER1$/check_udp -H $HOSTADDRESS$ -p $ARG1$
          }


# 'check_ftp' command definition
define command{
          command_name       check_ftp
          command_line       $USER1$/check_ftp -H $HOSTADDRESS$
          }


# 'check_pop' command definition
define command{
          command_name       check_pop
          command_line       $USER1$/check_pop -H $HOSTADDRESS$
          }


# 'check_smtp' command definition
define command{
          command_name       check_smtp
          command_line       $USER1$/check_smtp -H $HOSTADDRESS$
          }


# 'check_nntp' command definition
define command{
          command_name       check_nntp
          command_line       $USER1$/check_nntp -H $HOSTADDRESS$
          }
```

```
# 'check_http' command definition
define command{
          command_name       check_http
          command_line       $USER1$/check_http -H $HOSTADDRESS$
          }


# 'check_telnet' command definition
define command{
          command_name       check_telnet
          command_line       $USER1$/check_tcp -H $HOSTADDRESS$ -p 23
          }

# 'check_ssh' command definition
define command{
        command_name    check_ssh
        command_line    $USER1$/check_tcp -H $HOSTADDRESS$ -p 22
        }

# 'check_surecom_http' command definition
define command{
        command_name    check_surecom_http
        command_line    $USER1$/check_tcp -H $HOSTADDRESS$ -p 12345
        }

# 'check_ping' command definition
define command{
          command_name       check_ping
          command_line       $USER1$/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$ -p 5
          }


# 'check_dns' command definition
define command{
          command_name       check_dns
          command_line       $USER1$/check_dns -H www.yahoo.com -s $HOSTADDRESS$
          }

# 'check_hpjd' command definition
define command{
          command_name       check_hpjd
          command_line       $USER1$/check_hpjd -H $HOSTADDRESS$ -C public
          }


# 'check_local_disk' command definition
define command{
          command_name       check_local_disk
          command_line       $USER1$/check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$
          }


# 'check_local_users' command definition
define command{
          command_name       check_local_users
          command_line       $USER1$/check_users -w $ARG1$ -c $ARG2$
          }


# 'check_local_procs' command definition
define command{
          command_name       check_local_procs
          command_line       $USER1$/check_procs -w $ARG1$ -c $ARG2$ -s $ARG3$
          }


# 'check_local_load' command definition
define command{
          command_name       check_local_load
```

```
        command_line       $USER1$/check_load -w $ARG1$ -c $ARG2$
        }



################################################################################
#
# SAMPLE HOST CHECK COMMANDS
#
################################################################################


# This command checks to see if a host is "alive" by pinging it
# The check must result in a 100% packet loss or 5 second (5000ms) round trip
# average time to produce a critical error.
# Note: Only one ICMP echo packet is sent (determined by the '-p 1' argument)

# 'check-host-alive' command definition
define command{
        command_name    check-host-alive
        command_line    $USER1$/check_ping -H $HOSTADDRESS$ -w 3000.0,80% -c 5000.0,100% -
p 1
        }
```

```
##############################################################################
#
# CONTACTGROUPS.CFG
#
##############################################################################

# 'oriental-admins' contact group definition
define contactgroup{
          contactgroup_name  oriental-admins
          alias                         Oriental Nagios Administrators
          members                       davidmar
          }


# 'platao-admins' contact group definition
define contactgroup{
          contactgroup_name  platao-admins
          alias                         PLATAO Administrators
          members                       vmac
          }


# 'surecomwireless-admins' contact group definition
define contactgroup{
          contactgroup_name  surecomwireless-admins
          alias                         SURECOM WIRELESS Administrators
          members                       vmac
          }
```

```
##############################################################################
#
# CONTACTS.CFG
#
##############################################################################

# 'davidmar' contact definition
define contact{
          contact_name                            davidmar
          alias                                   Nagios Admin
          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                                   davidmar@netcabo.pt
          pager                                   davidmar@netcabo.pt
          }


# 'vmac' contact definition
define contact{
          contact_name                            vmac
          alias                                   Vasco Costa
          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                                   vmac@netcabo.pt
          }
```

```
##############################################################################
#
# DEPENDENCIES.CFG
#
##############################################################################


define hostdependency{
          host_name                     surecom
          dependent_host_name           platao
          notification_failure_criteria         d,u
          }

define hostdependency{
        host_name                       surecom
        dependent_host_name             surecomwireless
        notification_failure_criteria   d,u
        }

define hostdependency{
        host_name                       surecom
        dependent_host_name             mar
        notification_failure_criteria   d,u
        }
```

```
##############################################################################
#
# HOSTEXTINFO.CFG - Ficheiro de Definições de Informalção Extendida para Hosts
#
##############################################################################


define hostextinfo{
         host_name          surecom
         icon_image                 surecom.png
         icon_image_alt             SURECOM ROUTER EP4504 AX
         vrml_image                 surecom.png
         }


define hostextinfo{
        host_name               oriental
        icon_image              oriental.jpg
        icon_image_alt          ORIENTAL ROUTER & SERVER
        vrml_image              oriental.jpg
        }


define hostextinfo{
        host_name               241s1
        icon_image              241s1.png
        icon_image_alt          241S1 FOSA/AIRIS
        vrml_image              241s1.png
        }
```

```
##############################################################################
#
# HOSTGROUPS.CFG
#
##############################################################################

# 'oriental-servers' host group definition
define hostgroup{
         hostgroup_name     oriental-servers
         alias              ORIENTAL UNDER SERVERS
         contact_groups     oriental-admins
         members            oriental, surecom, mar, 241s1
         }


# 'platao-servers' host group definition
define hostgroup{
         hostgroup_name     platao-servers
         alias              PLATAO Server
         contact_groups     platao-admins
         members            platao
         }


# 'surecomwireless-boxes' host group definition
define hostgroup{
         hostgroup_name     surecomwireless-servers
         alias              SURECOM WIRELESS UNDER SERVERS
         contact_groups     surecomwireless-admins
         members            surecomwireless
         }
```

```
##############################################################################
#
# HOSTS.CFG
#
##############################################################################

# Generic host definition template
define host{
         name                                    generic-host       ; The name of this host
template - referenced in other host definitions, used for template recursion/resolution
         notifications_enabled                   1         ; Host notifications are enabled
         event_handler_enabled                   1         ; Host event handler is enabled
         flap_detection_enabled                  1         ; Flap detection is enabled
         process_perf_data            1         ; Process performance data
         retain_status_information    1         ; Retain status information across program
restarts
         retain_nonstatus_information 1         ;  Retain  non-status  information  across
program restarts

         register                     0         ; DONT REGISTER THIS DEFINITION - ITS NOT
A REAL HOST, JUST A TEMPLATE!
         }

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

        host_name                surecom
        alias                    SURECOM ROUTER EP4504 AX
        address                  192.168.1.1
        check_command            check-host-alive
        max_check_attempts       10
        notification_interval    120
        notification_period      24x7
        notification_options     d,u,r
        }



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

         host_name           oriental
         alias                         ORIENTAL ROUTER & SERVER
         address                       192.168.1.4
         parents                       surecom
         check_command                 check-host-alive
         max_check_attempts 10
         notification_interval         120
         notification_period24x7
         notification_options          d,u,r
         }


# '241s1' host definition
define host{

         host_name           241s1
         alias                         241S1 FOSA/AIRIS Lap Top
         address                       192.168.0.4
         parents                       oriental
         check_command                 check-host-alive
         max_check_attempts 10
         notification_interval         120
         notification_period24x7
         notification_options          d,u,r
         }
```

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

        host_name           mar
        alias                           MAR Desktop
        address                         192.168.1.3
        parents                         surecom
        check_command                   check-host-alive
        max_check_attempts 10
        notification_interval           120
        notification_period24x7
        notification_options            d,u,r
        }


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

        host_name           platao
        alias                           PLATAO SERVER - Máquina de Alunos do ISCTE
        address                         193.136.191.98
        check_command                   check-host-alive
        max_check_attempts 10
        notification_interval           480
        notification_period24x7
        notification_options            d,u,r
        }


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

        host_name           surecomwireless
        alias                           SURECOM WIRELESS ROUTER
        address                         unixed.net
        check_command                   check-host-alive
        max_check_attempts 10
        notification_interval           480
        notification_period24x7
        notification_options            d,u,r
        }
```

```
##############################################################################
#
# MISCCOMMANDS.CFG
#
##############################################################################


##############################################################################
# COMMAND DEFINITIONS
#
# SYNTAX:
#
#         define command{
#                template      <templatename>
#                   name          <objectname>
#                command_name  <commandname>
#                command_line  <commandline>
#                }
#
# WHERE:
#
# <templatename> = object name of another command definition that should be
#                  used as a template for this definition (optional)
# <objectname>   = object name of command definition, referenced by other
#                  command definitions that use it as a template (optional)
# <commandname>  = name of the command, as recognized/used by Nagios
# <commandline>  = command line
#
##############################################################################


# 'notify-by-email' command definition
define command{
          command_name       notify-by-email
          command_line       /usr/bin/printf "%b" "***** Nagios 1.0 *****\n\nNotification
Type:     $NOTIFICATIONTYPE$\n\nService:     $SERVICEDESC$\nHost:     $HOSTALIAS$\nAddress:
$HOSTADDRESS$\nState:         $SERVICESTATE$\n\nDate/Time:         $DATETIME$\n\nAdditional
Info:\n\n$OUTPUT$"    |    /usr/bin/mail    -s    "**    $NOTIFICATIONTYPE$    alert    -
$HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **" $CONTACTEMAIL$
          }


# 'notify-by-epager' command definition
define command{
          command_name       notify-by-epager
          command_line       /usr/bin/printf     "%b"     "Service:     $SERVICEDESC$\nHost:
$HOSTNAME$\nAddress:    $HOSTADDRESS$\nState:    $SERVICESTATE$\nInfo:    $OUTPUT$\nDate:
$DATETIME$"   |   /usr/bin/mail   -s   "$NOTIFICATIONTYPE$:   $HOSTALIAS$/$SERVICEDESC$   is
$SERVICESTATE$" $CONTACTPAGER$
          }


# 'host-notify-by-email' command definition
define command{
          command_name       host-notify-by-email
          command_line       /usr/bin/printf "%b" "***** Nagios 1.0 *****\n\nNotification
Type:       $NOTIFICATIONTYPE$\nHost:       $HOSTNAME$\nState:       $HOSTSTATE$\nAddress:
$HOSTADDRESS$\nInfo:   $OUTPUT$\n\nDate/Time:   $DATETIME$\n"   |   /usr/bin/mail   -s   "Host
$HOSTSTATE$ alert for $HOSTNAME$!" $CONTACTEMAIL$
          }


# 'host-notify-by-epager' command definition
define command{
          command_name       host-notify-by-epager
          command_line       /usr/bin/printf      "%b"      "Host      '$HOSTALIAS$'      is
$HOSTSTATE$\nInfo:  $OUTPUT$\nTime:  $DATETIME$"  |  /usr/bin/mail  -s  "$NOTIFICATIONTYPE$
alert - Host $HOSTNAME$ is $HOSTSTATE$" $CONTACTPAGER$
          }
```

```
##############################################################################
#
# PERFORMANCE DATA COMMANDS
#
# These are sample performance data commands that can be used to send performance
# data output to two text files (one for hosts, another for services).  If you
# plan on simply writing performance data out to a file, consider compiling
# Nagios with native file support for performance data.  This is done by
# supplying the --with-file-perfdata option to the configure script.
#
##############################################################################


# 'process-host-perfdata' command definition
define command{
         command_name       process-host-perfdata
         command_line       /usr/bin/printf                                                 "%b"
"$LASTCHECK$\t$HOSTNAME$\t$HOSTSTATE$\t$HOSTATTEMPT$\t$STATETYPE$\t$EXECUTIONTIME$\t$OUTPU
T$\t$PERFDATA$" >> /var/spool/nagios/host-perfdata.out
         }


# 'process-service-perfdata' command definition
define command{
         command_name       process-service-perfdata
         command_line       /usr/bin/printf                                                 "%b"
"$LASTCHECK$\t$HOSTNAME$\t$SERVICEDESC$\t$SERVICESTATE$\t$SERVICEATTEMPT$\t$STATETYPE$\t$E
XECUTIONTIME$\t$LATENCY$\t$OUTPUT$\t$PERFDATA$" >> /var/spool/nagios/service-perfdata.out
         }
```

```
##############################################################################
#
# RESOURCE.CFG
#
##############################################################################

##############################################################################
#
# You can define $USERx$ macros in this file, which can in turn be used
# in command definitions in your host config file(s).  $USERx$ macros are
# useful for storing sensitive information such as usernames, passwords,
# etc.  They are also handy for specifying the path to plugins and
# event handlers - if you decide to move the plugins or event handlers to
# a different directory in the future, you can just update one or two
# $USERx$ macros, instead of modifying a lot of command definitions.
#
# The CGIs will not attempt to read the contents of resource files, so
# you can set restrictive permissions (600 or 660) on them.
#
# Nagios supports up to 32 $USERx$ macros ($USER1$ through $USER32$)
#
# Resource files may also be used to store configuration directives for
# external data sources like MySQL...
#
##############################################################################

# Sets $USER1$ to be the path to the plugins
$USER1$=/usr/local/libexec/nagios

# Sets $USER2$ to be the path to event handlers
#$USER2$=/usr/local/libexec/nagios/eventhandlers

# Store some usernames and passwords (hidden from the CGIs)
#$USER3$=someuser
#$USER4$=somepassword


# DB STATUS DATA
# Note: These config directives are only used if you compiled
# in database support for status data!
# The user you specify here needs SELECT, INSERT, UPDATE, and
# DELETE privileges on the 'programstatus', 'hoststatus',
# and 'servicestatus' tables in the database.

#xsddb_host=somehost
#xsddb_port=someport
#xsddb_database=somedatabase
#xsddb_username=someuser
#xsddb_password=somepassword
#xsddb_optimize_data=1
#xsddb_optimize_interval=3600


# DB COMMENT DATA
# Note: These config directives are only used if you compiled
# in database support for comment data!
# The user you specify here needs SELECT, INSERT, UPDATE, and
# DELETE privileges on the 'hostcomments' and 'servicecomments'
# tables in the database.

#xcddb_host=somehost
#xcddb_port=someport
#xcddb_database=somedatabase
#xcddb_username=someuser
#xcddb_password=somepassword
#xcddb_optimize_data=1



# DB DOWNTIME DATA
# Note: These config directives are only used if you compiled
```

```
# in database support for downtime data!
# The user you specify here needs SELECT, INSERT, UPDATE, and
# DELETE privileges on the 'hostdowntime' and 'servicedowntime'
# tables in the database.

#xdddb_host=somehost
#xdddb_port=someport
#xdddb_database=somedatabase
#xdddb_username=someuser
#xdddb_password=somepassword
#xdddb_optimize_data=1


# DB RETENTION DATA
# Note: These config directives are only used if you compiled
# in database support for retention data!
# The user you specify here needs SELECT, INSERT, UPDATE, and
# DELETE privileges on the 'programretention', 'hostretention',
# and 'serviceretention' tables in the database.

#xrddb_host=somehost
#xrddb_port=someport
#xrddb_database=somedatabase
#xrddb_username=someuser
#xrddb_password=somepassword
#xrddb_optimize_data=1
```

```
##############################################################################
#
# SERVICEEXTINFO.CFG
#
##############################################################################



# A simple definition - applies to a single service (on a single host)

define serviceextinfo{
          host_name           host9
          service_description          PING
          icon_image                   ping.gif
          }



# This definition is applied to several services on different hosts.
# In order for this to work, all services need to be named the same way.
# In this case, all the services are named 'TCP Wrappers'

define serviceextinfo{
          name sei1
          host_name           host1,host2,host3,host4
          service_description          TCP Wrappers
          icon_image                   wrappers.gif
          }



# This definition will also be applied to several services on different
# hosts because the 'host_name' member will be inherited from the
# definition above.

define serviceextinfo{
          use sei1
          service_description          Security Alerts
          icon_image                   security.gif
          }



# This definition is applied to all hosts in a specific hostgroup.
# You can have it apply to all hosts in multiple hostgroups by
# separating different hostgroups with commas...

define serviceextinfo{
          hostgroup           novell-servers
          service_descriptionLRU Sitting Time
          icon_image                   cache.gif
        }
```

```
##############################################################################
#
# SERVICES.CFG
#
##############################################################################

# Generic service definition template
define service{
         name                                    generic-service    ; The 'name' of this
service template, referenced in other service definitions
         active_checks_enabled                   1        ;  Active  service  checks  are
enabled
         passive_checks_enabled                  1        ;  Passive  service  checks  are
enabled/accepted
         parallelize_check           1        ;  Active  service  checks  should  be
parallelized (disabling this can lead to major performance problems)
         obsess_over_service         1        ; We should obsess over this service (if
necessary)
         check_freshness                         0        ;  Default  is  to  NOT  check
service 'freshness'
         notifications_enabled                   1        ;  Service  notifications  are
enabled
         event_handler_enabled                   1        ;  Service  event  handler  is
enabled
         flap_detection_enabled                  1        ; Flap detection is enabled
         process_perf_data           1        ; Process performance data
         retain_status_information   1        ; Retain status information across program
restarts
         retain_nonstatus_information 1       ;  Retain  non-status  information  across
program restarts

         register                    0        ; DONT REGISTER THIS DEFINITION - ITS NOT
A REAL SERVICE, JUST A TEMPLATE!
         }

# Service definition for checking system load
define service{

         use                generic-service

         service_description         System Load
         host_name                   oriental
         contact_groups              oriental-admins
         notification_options        w,c,r
         is_volatile                    0
        check_period                  24x7
        max_check_attempts            3
        normal_check_interval         3
        retry_check_interval          1
         notification_interval          120
        notification_period           24x7

         check_command check_local_load!1.00,1.25,1.50!2.00,2.00,2.00

}


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

         host_name                   oriental
         service_description         HTTP
         is_volatile                             0
         check_period                            24x7
         max_check_attempts          3
         normal_check_interval                   3
         retry_check_interval                    1
         contact_groups                          oriental-admins
         notification_interval                   120
```

```
        notification_period         24x7
        notification_options                w,u,c,r
        check_command                       check_http
        }


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

        host_name                   oriental
        service_description         FTP
        is_volatile                         0
        check_period                        24x7
        max_check_attempts          3
        normal_check_interval               5
        retry_check_interval                1
        contact_groups                      oriental-admins
        notification_interval               120
        notification_period         24x7
        notification_options                w,u,c,r
        check_command                       check_ftp
        }

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

       host_name                      oriental
       service_description            SSH
       is_volatile                    0
       check_period                   24x7
       max_check_attempts             3
       normal_check_interval          3
       retry_check_interval           1
       contact_groups                 oriental-admins
       notification_interval          120
       notification_period            24x7
       notification_options           w,u,c,r
       check_command                  check_ssh
       }

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

       host_name                      oriental
       service_description            /dev/ad0s1e Free Space
       is_volatile                    0
       check_period                   24x7
       max_check_attempts             3
       normal_check_interval          3
       retry_check_interval           1
       contact_groups                 oriental-admins
       notification_interval          120
       notification_period            24x7
       notification_options           w,u,c,r
       check_command                  check_local_disk!20%!10%!/dev/ad0s1e
       }

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

        host_name                   surecom
        service_description         PING
        is_volatile                         0
```

```
        check_period                    24x7
        max_check_attempts      3
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  oriental-admins
        notification_interval           120
        notification_period     24x7
        notification_options            c,r
        check_command                   check_ping!100.0,20%!500.0,60%
        }


# Service definition
define service{
        use                             generic-service

        host_name                       surecom
        service_description             SURECOM HTTP
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  oriental-admins
        notification_interval           120
        notification_period             24x7
        notification_options            c,r
        check_command                   check_surecom_http
        }


define service{

        use                             generic-service

        host_name                       platao
        service_description             HTTP
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  platao-admins
        notification_interval           240
        notification_period             24x7
        notification_options            w,u,c,r
        check_command                   check_http
        }

define service{

        use                             generic-service

        host_name                       platao
        service_description             SSH
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  platao-admins
        notification_interval           240
        notification_period             24x7
        notification_options            w,u,c,r
        check_command                   check_ssh
        }

define service{

        use                             generic-service
```

```
        host_name                       platao
        service_description             FTP
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  platao-admins
        notification_interval           240
        notification_period             24x7
        notification_options            w,u,c,r
        check_command                   check_ftp
        }


# Service definition
define service{
         use                                    generic-service

         host_name                      surecomwireless
         service_description            PING
         is_volatile                            0
         check_period                           24x7
         max_check_attempts             3
         normal_check_interval                  5
         retry_check_interval                   1
         contact_groups                         surecomwireless-admins
         notification_interval                  120
         notification_period            24x7
         notification_options                   c,r
         check_command                          check_ping!100.0,20%!500.0,60%
         }

# Service definition
define service{
        use                             generic-service

        host_name                       241s1
        service_description             PING
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                  oriental-admins
        notification_interval           120
        notification_period             24x7
        notification_options            c,r
        check_command                   check_ping!100.0,20%!500.0,60%
        }


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

        host_name                       mar
        service_description             PING
        is_volatile                     0
        check_period                    24x7
        max_check_attempts              3
        normal_check_interval           5
        retry_check_interval            1
        contact_groups                          oriental-admins
          notification_interval           120
        notification_period             24x7
        notification_options            c,r
        check_command                   check_ping!100.0,20%!500.0,60%
        }
```

```
##############################################################################
#
# TIMEPERIODS.CFG
#
##############################################################################

# '24x7' timeperiod definition
define timeperiod{
          timeperiod_name    24x7
          alias              24 Hours A Day, 7 Days A Week
          sunday             00:00-24:00
          monday             00:00-24:00
          tuesday            00:00-24:00
          wednesday 00:00-24:00
          thursday  00:00-24:00
          friday             00:00-24:00
          saturday  00:00-24:00
          }


# 'workhours' timeperiod definition
define timeperiod{
          timeperiod_name    workhours
          alias              "Normal" Working Hours
          monday             09:00-17:00
          tuesday            09:00-17:00
          wednesday 09:00-17:00
          thursday  09:00-17:00
          friday             09:00-17:00
          }


# 'nonworkhours' timeperiod definition
define timeperiod{
          timeperiod_name    nonworkhours
          alias              Non-Work Hours
          sunday             00:00-24:00
          monday             00:00-09:00,17:00-24:00
          tuesday            00:00-09:00,17:00-24:00
          wednesday 00:00-09:00,17:00-24:00
          thursday  00:00-09:00,17:00-24:00
          friday             00:00-09:00,17:00-24:00
          saturday  00:00-24:00
          }


# 'none' timeperiod definition
define timeperiod{
          timeperiod_name    none
          alias              No Time Is A Good Time
          }
```

```
################################################################################
#
# HTPASSWD.USERS - Ficheiro de configuração para Utilizadores do Nagios em Apache
#
################################################################################

guru:********************
davidmar:******************
vmac:**********************
```

ANEXO B – *Impressões do GUI do Nagios*

Tactical Overview

Tactical Monitoring Overview
Last Updated: Sun Nov 16 19:22:19 WET 2003
Updated every 90 seconds
Nagios® - www.nagios.org
Logged in as *davidmar*

Monitoring Performance

Check Execution Time:	0 / 10 / 2.692 sec
Check Latency:	0 / 0 / 0.000 sec
# Active Checks:	13
# Passive Checks:	0

Network Outages

N/A

Network Health

Host Health:	
Service Health:	

Hosts

4 Down	0 Unreachable	2 Up	0 Pending
4 Unhandled Problems			

Services

3 Critical	0 Warning	0 Unknown	10 Ok	0 Pending
3 on Problem Hosts				

Monitoring Features

Flap Detection	Notifications	Event Handlers	Active Checks	Passive Checks
Disabled	Enabled	Enabled	Enabled	Enabled
N/A	1 Service Disabled 3 Hosts Disabled	All Services Enabled All Hosts Enabled	All Services Enabled All Hosts Enabled	All Services Enabled

Service Detail

Service Status Details
For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
241s1	PING	CRITICAL	16-11-2003 19:25:16	0d 17h 50m 5s	1/3	CRITICAL - Plugin timed out after 10 seconds
mar	PING	OK	16-11-2003 19:23:59	0d 0h 2m 26s	1/3	PING OK - Packet loss = 0%, RTA = 0.45 ms
oriental	/dev/ad0s1e Free Space	OK	16-11-2003 19:23:56	3d 21h 35m 51s	1/3	DISK OK [722864 kB (33%) free on /dev/ad0s1e]
	FTP	OK	16-11-2003 19:23:58	5d 19h 51m 20s	1/3	FTP OK - 0.056 second response time on port 21 [220 oriental.medicinachinesa.com FTP server (Version 6.00LS) ready.]
	HTTP	OK	16-11-2003 19:23:56	5d 19h 48m 6s	1/3	HTTP ok: HTTP/1.1 200 OK - 0.021 second response time
	SSH	OK	16-11-2003 19:23:56	0d 19h 1m 10s	1/3	TCP OK - 0.002 second response time on port 22
	System Load	OK	16-11-2003 19:23:56	4d 21h 44m 29s	1/3	OK - load average: 0.00, 0.00, 0.00
platao	FTP	OK	16-11-2003 19:24:41	0d 3h 11m 56s	1/3	FTP OK - 4.290 second response time on port 21 [220 ProFTPD 1.2.9rc2 Server (platao) [alunos.iscte.pt]]
	HTTP	OK	16-11-2003 19:23:59	0d 1h 22m 26s	1/3	HTTP ok: HTTP/1.1 200 OK - 0.755 second response time
	SSH	OK	16-11-2003 19:23:59	0d 19h 1m 7s	1/3	TCP OK - 0.249 second response time on port 22
surecom	PING	OK	16-11-2003 19:23:59	5d 19h 47m 32s	1/3	PING OK - Packet loss = 0%, RTA = 0.94 ms
	SURECOM HTTP	OK	16-11-2003 19:24:26	0d 19h 0m 34s	1/3	TCP OK - 0.013 second response time on port 12345
surecomwireless	PING	CRITICAL	16-11-2003 19:21:56	0d 0h 47m 55s	1/3	CRITICAL - Plugin timed out after 10 seconds

13 Matching Service Entries Displayed

Host Detail

Host Status Details For All Host Groups

Host	Status	Last Check	Duration	Status Information
241s1	DOWN	16-11-2003 19:30:16	0d 17h 55m 40s	CRITICAL - Plugin timed out after 10 seconds
mar	UP	16-11-2003 19:24:25	0d 0h 8m 1s	PING OK - Packet loss = 0%, RTA = 0.46 ms
oriental	UP	16-11-2003 01:36:46	5d 19h 57m 20s	PING OK - Packet loss = 0%, RTA = 0.23 ms
platao	DOWN	16-11-2003 19:29:55	0d 16h 55m 51s	CRITICAL - Plugin timed out after 10 seconds
surecom	UP	16-11-2003 00:47:27	5d 19h 53m 7s	(Host assumed to be up)
surecomwireless	DOWN	16-11-2003 19:31:56	0d 0h 53m 30s	CRITICAL - Plugin timed out after 10 seconds

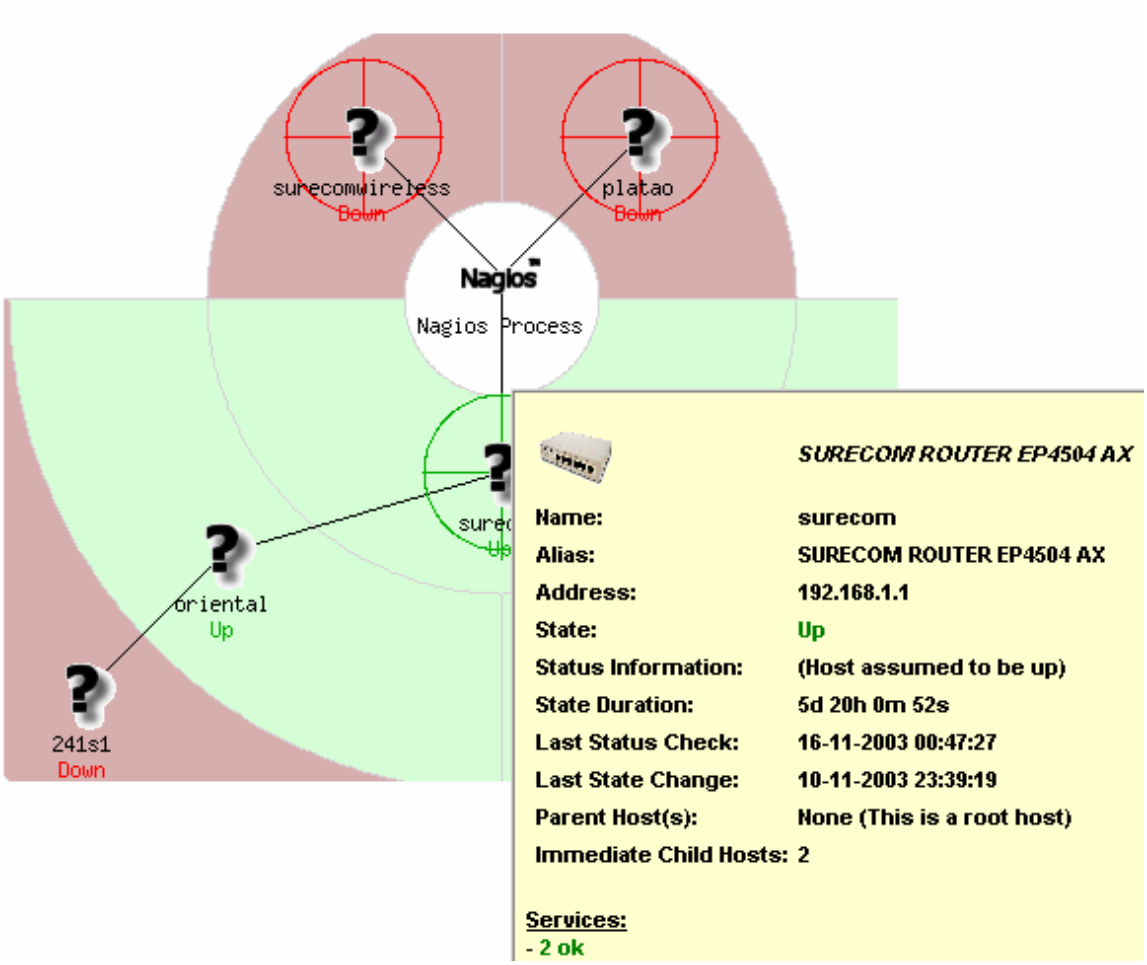
6 Matching Host Entries Displayed

Status Summary

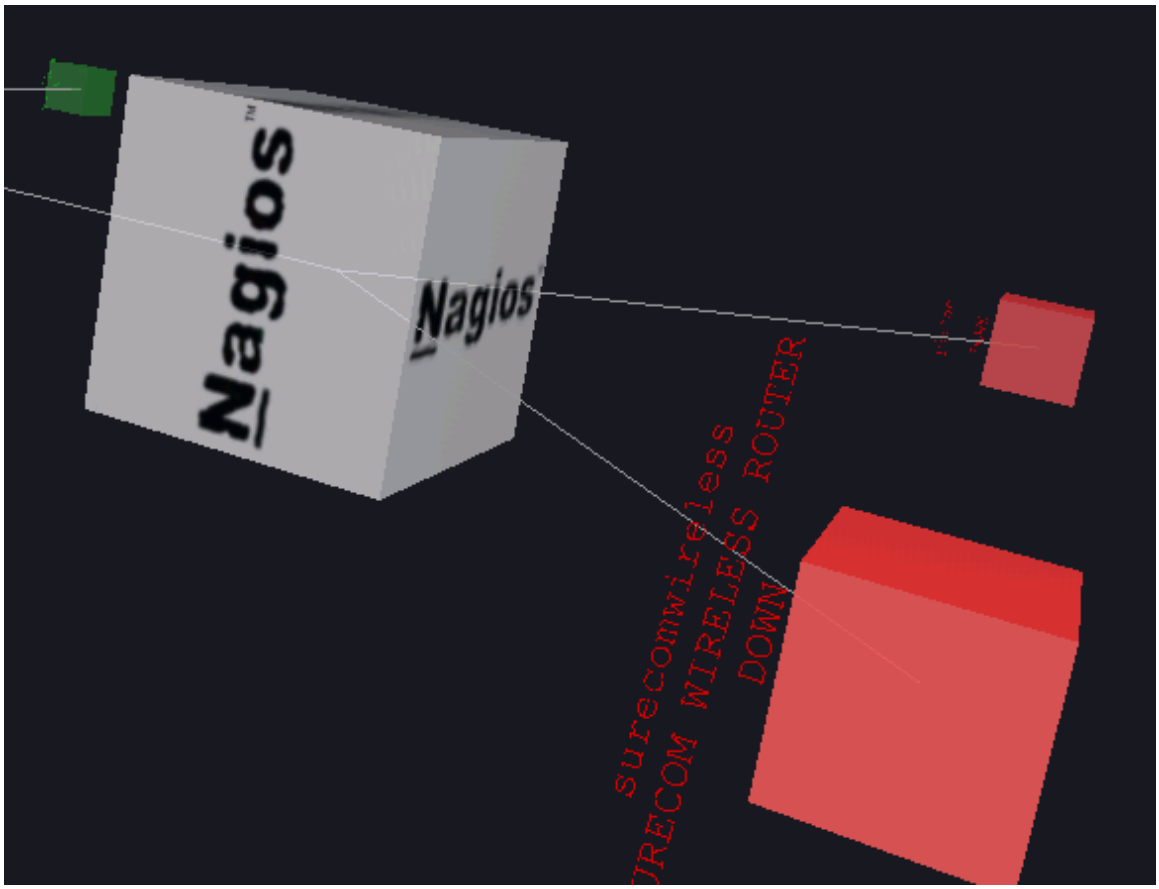
Status Summary For All Host Groups

Host Group	Host Status Totals	Service Status Totals
ORIENTAL UNDER SERVERS (oriental-servers)	3 UP 1 DOWN	8 OK 1 CRITICAL
PLATAO Server (platao-servers)	1 DOWN	3 OK
SURECOM WIRELESS UNDER SERVERS (surecomwireless-servers)	1 DOWN	1 CRITICAL

Status Map


surecomwireless
Down
platao
Down
Nagios
Nagios Process
oriental
Up
241s1
Down
SURECOM ROUTER EP4504 AX
Name: surecom
Alias: SURECOM ROUTER EP4504 AX
Address: 192.168.1.1
State: Up
Status Information: (Host assumed to be up)
State Duration: 5d 20h 0m 52s
Last Status Check: 16-11-2003 00:47:27
Last State Change: 10-11-2003 23:39:19
Parent Host(s): None (This is a root host)
Immediate Child Hosts: 2
Services:
- 2 ok

3-D Status Map



Service Problems

Service Status Details For All Hosts

Host	Service	Status	Last Check	Duration	Attempt	Status Information
241s1	PING	CRITICAL	16-11-2003 20:00:16	0d 18h 25m 27s	1/3	CRITICAL - Plugin timed out after 10 seconds
surecomwireless	PING	CRITICAL	16-11-2003 19:56:56	0d 1h 23m 17s	1/3	CRITICAL - Plugin timed out after 10 seconds

2 Matching Service Entries Displayed

Host Problems

Host Status Details For All Host Groups

Host	Status	Last Check	Duration	Status Information
241s1	DOWN	16-11-2003 20:00:16	0d 18h 26m 58s	CRITICAL - Plugin timed out after 10 seconds
platao	DOWN	16-11-2003 19:59:45	0d 17h 27m 9s	CRITICAL - Plugin timed out after 10 seconds
surecomwireless	DOWN	16-11-2003 20:01:56	0d 1h 24m 48s	CRITICAL - Plugin timed out after 10 seconds

3 Matching Host Entries Displayed

Comments

Host Comments

Add a new host comment

Host Name	Entry Time	Author	Comment	Comment ID	Persistent	Actions
platao	12-11-2003 21:52:40	David Mar	ACKNOWLEDGEMENT: Não Responde a Pings	1	Yes	

Service Comments

Add a new service comment

Host Name	Service	Entry Time	Author	Comment	Comment ID	Persistent	Actions
There are no service comments							

Process Info

Process Information

Program Start Time:	16-11-2003 00:44:25
Total Running Time:	0d 19h 22m 37s
Last External Command Check:	16-11-2003 20:06:54
Last Log File Rotation:	N/A
Nagios PID	29927
Notifications Enabled?	YES
Service Checks Being Executed?	YES
Passive Service Checks Being Accepted?	YES
Event Handlers Enabled?	Yes
Obsessing Over Services?	No
Flap Detection Enabled?	No
Performance Data Being Processed?	No

Process Commands

- Shutdown the Nagios process
- Restart the Nagios process
- Disable notifications
- Stop executing service checks
- Stop accepting passive service checks
- Disable event handlers
- Start obsessing over services
- Enable flap detection
- Enable performance data

Process Status Information

Process Status:	OK
Check Command Output:	No process check command has been defined in your CGI configuration file. Nagios process is assumed to be running properly.

Performance Info

Program-Wide Performance Information

Active Checks:

Time Frame	Checks Completed
<= 1 minute:	1 (7.7%)
<= 5 minutes:	13 (100.0%)
<= 15 minutes:	13 (100.0%)
<= 1 hour:	13 (100.0%)
Since program start:	13 (100.0%)

Metric	Min.	Max.	Average
Check Execution Time:	< 1 sec	11 sec	3.077 sec
Check Latency:	< 1 sec	< 1 sec	0.000 sec
Percent State Change:	0.00%	0.00%	0.00%

Passive Checks:

Time Frame	Checks Completed
<= 1 minute:	0 (0.0%)
<= 5 minutes:	0 (0.0%)
<= 15 minutes:	0 (0.0%)
<= 1 hour:	0 (0.0%)
Since program start:	0 (0.0%)

Metric	Min.	Max.	Average
Percent State Change:	0.00%	0.00%	0.00%

Scheduling Queue

Entries sorted by **next check time** (ascending)

Host	Service	Last Check	Next Check	Active Checks	Actions
surecomwireless	PING	16-11-2003 20:06:56	16-11-2003 20:11:56	ENABLED	
oriental	/dev/ad0s1e Free Space	16-11-2003 20:09:25	16-11-2003 20:12:25	ENABLED	
oriental	HTTP	16-11-2003 20:09:25	16-11-2003 20:12:25	ENABLED	
oriental	SSH	16-11-2003 20:09:25	16-11-2003 20:12:25	ENABLED	
oriental	System Load	16-11-2003 20:09:25	16-11-2003 20:12:25	ENABLED	
mar	PING	16-11-2003 20:08:59	16-11-2003 20:13:59	ENABLED	
oriental	FTP	16-11-2003 20:08:59	16-11-2003 20:13:59	ENABLED	
platao	HTTP	16-11-2003 20:08:59	16-11-2003 20:13:59	ENABLED	
platao	SSH	16-11-2003 20:08:59	16-11-2003 20:13:59	ENABLED	
surecom	PING	16-11-2003 20:08:59	16-11-2003 20:13:59	ENABLED	
surecom	SURECOM HTTP	16-11-2003 20:09:26	16-11-2003 20:14:26	ENABLED	
platao	FTP	16-11-2003 20:09:41	16-11-2003 20:14:41	ENABLED	
241s1	PING	16-11-2003 20:10:16	16-11-2003 20:15:16	ENABLED	

Availability

All Hosts

15-11-2003 20:13:57 to 16-11-2003 20:13:57
Duration: 1d 0h 0m 0s

Assume initial states: yes

Assume state retention: yes

First assumed host state: Unspecified

Backtracked archives: 4

Report period: [Current time range]

Update

[Availability report completed in 0 min 0 sec]

Host State Breakdowns:

Host	% Time Up	% Time Down	% Time Unreachable	% Time Undetermined
241s1	22.418% (22.418%)	77.582% (77.582%)	0.000% (0.000%)	0.000%
mar	21.808% (21.808%)	78.192% (78.192%)	0.000% (0.000%)	0.000%
oriental	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	100.000%
platao	0.000% (0.000%)	100.000% (100.000%)	0.000% (0.000%)	0.000%
surecom	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	100.000%
surecomwireless	3.152% (3.152%)	96.848% (96.848%)	0.000% (0.000%)	0.000%

Availability

All Services

15-11-2003 00:00:00 to 16-11-2003 00:00:00
Duration: 1d 0h 0m 0s

Assume initial states: yes

Assume state retention: yes

First assumed service state: Unspecified

Backtracked archives: 4

Report period: [Current time range]

Update

[Availability report completed in 0 min 0 sec]

Service State Breakdowns:

Host	Service	% Time OK	% Time Warning	% Time Unknown	% Time Critical	% Time Undetermined
241s1	PING	62.270% (62.270%)	0.000% (0.000%)	0.000% (0.000%)	37.730% (37.730%)	0.000%
mar	PING	19.919% (19.919%)	0.000% (0.000%)	0.000% (0.000%)	80.081% (80.081%)	0.000%
oriental	/dev/ad0s1e Free Space	100.000% (100.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000%
	FTP	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	100.000%
	HTTP	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	100.000%
	SSH	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	100.000%
	System Load	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	100.000%
platao	FTP	78.704% (78.704%)	0.000% (0.000%)	0.000% (0.000%)	21.296% (21.296%)	0.000%
	HTTP	80.183% (80.183%)	0.000% (0.000%)	0.000% (0.000%)	19.817% (19.817%)	0.000%
	SSH	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	100.000%
surecom	PING	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	100.000%
	SURECOM HTTP	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	0.000% (0.000%)	100.000%
surecomwireless	PING	3.265% (3.265%)	0.588% (0.588%)	0.000% (0.000%)	96.147% (96.147%)	0.000%

Alert History

Latest Archive

Log File Navigation
Sun Nov 16 00:00:00 WET 2003 to Present..

File: /var/spool/nagios/nagios.log

November 16, 2003 19:00

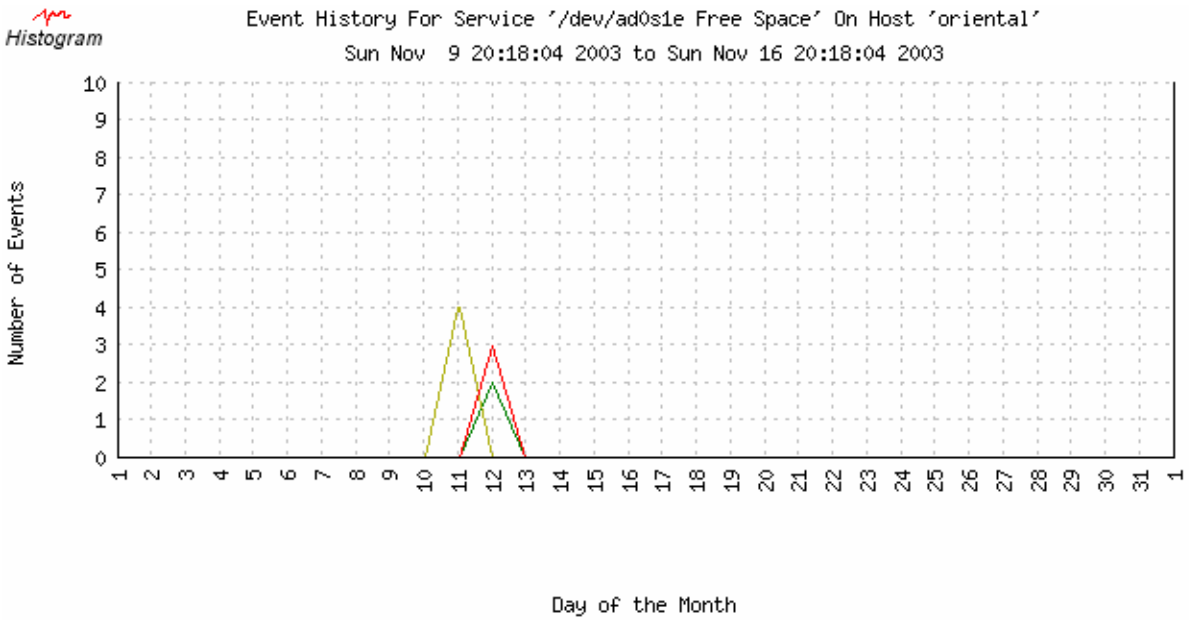
[16-11-2003 19:44:25] SERVICE ALERT: platao;HTTP;OK;HARD;1;HTTP ok: HTTP/1.1 200 OK - 6.704 second response time

[16-11-2003 19:39:25] SERVICE ALERT: platao;HTTP;CRITICAL;HARD;1;Socket timeout after 10 seconds

[16-11-2003 19:24:25] SERVICE ALERT: mar;PING;OK;HARD;1;PING OK - Packet loss = 0%, RTA = 0.45 ms

[16-11-2003 19:24:25] HOST ALERT: mar;UP;HARD;1;PING OK - Packet loss = 0%, RTA = 0.46 ms

Alert Histogram



EVENT TYPE	MIN	MAX	SUM	AVG
Recovery (Ok):	0	2	2	0.06
Warning:	0	4	4	0.13
Unknown:	0	0	0	0.00
Critical:	0	3	3	0.10

Notifications

All Contacts

Latest Archive

Log File Navigation
Sun Nov 16 00:00:00 WET 2003
to
Present..

File: /var/spool/nagios/nagios.log

Notification detail level for all contacts:

All notifications

Older Entries First:

Update

Host	Service	Type	Time	Contact	Notification Command	Information
surecomwireless	PING	CRITICAL	16-11-2003 15:16:07	davidmar	notify-by-email	PING CRITICAL - Packet loss = 60%, RTA = 80.49 ms
surecomwireless	PING	CRITICAL	16-11-2003 15:16:05	vmac	notify-by-email	PING CRITICAL - Packet loss = 60%, RTA = 80.49 ms
surecomwireless	PING	OK	16-11-2003 15:09:07	davidmar	notify-by-email	PING OK - Packet loss = 0%, RTA = 75.69 ms
surecomwireless	PING	OK	16-11-2003 15:09:05	vmac	notify-by-email	PING OK - Packet loss = 0%, RTA = 75.69 ms
platao	N/A	HOST DOWN	16-11-2003 12:14:06	davidmar	host-notify-by-email	CRITICAL - Plugin timed out after 10 seconds
platao	N/A	HOST DOWN	16-11-2003 12:14:05	vmac	host-notify-by-email	CRITICAL - Plugin timed out after 10 seconds
platao	N/A	HOST DOWN	16-11-2003 02:36:36	davidmar	host-notify-by-email	CRITICAL - Plugin timed out after 10 seconds
platao	N/A	HOST DOWN	16-11-2003 02:36:35	vmac	host-notify-by-email	CRITICAL - Plugin timed out after 10 seconds

View Config

Hosts

Host Name	Alias/Description	Address	Parent Hosts	Notification Interval	Notification Options	Notification Period	Max. Check Attempts	Host Check Command	Enable Checks	Event Handler	Enable Event Handler	Stalking Options	Enable Flap Detection	Low Flap Threshold	High Flap Threshold	Process Performance Data	Enable Failure Prediction	Failure Prediction Options	Retention Options
241s1	241S1 FOSA/AIRIS Lap Top	192.168.0.4	oriental	2h 0m 0s	Down, Unreachable, Recovery	24x7	10	check-host-alive	Yes		Yes	None	Yes	Program-wide value	Program-wide value	Yes	Yes		Status Information, Non-Status Information
mar	MAR Desktop	192.168.1.3	surecom	2h 0m 0s	Down, Unreachable, Recovery	24x7	10	check-host-alive	Yes		Yes	None	Yes	Program-wide value	Program-wide value	Yes	Yes		Status Information, Non-Status Information
oriental	ORIENTAL ROUTER & SERVER	192.168.1.4	surecom	2h 0m 0s	Down, Unreachable, Recovery	24x7	10	check-host-alive	Yes		Yes	None	Yes	Program-wide value	Program-wide value	Yes	Yes		Status Information, Non-Status Information
platao	PLATAO SERVER - Máquina de Alunos do ISCTE	193.136.191.98		8h 0m 0s	Down, Unreachable, Recovery	24x7	10	check-host-alive	Yes		Yes	None	Yes	Program-wide value	Program-wide value	Yes	Yes		Status Information, Non-Status Information
surecom	SURECOM ROUTER EP4504 AX	192.168.1.1		2h 0m 0s	Down, Unreachable, Recovery	24x7	10	check-host-alive	Yes		Yes	None	Yes	Program-wide value	Program-wide value	Yes	Yes		Status Information, Non-Status Information
surecomwireless	SURECOM WIRELESS ROUTER	unixed.net		8h 0m 0s	Down, Unreachable, Recovery	24x7	10	check-host-alive	Yes		Yes	None	Yes	Program-wide value	Program-wide value	Yes	Yes		Status Information, Non-Status Information

Host Dependencies

Dependent Host	Master Host	Dependency Type	Dependency Failure Options
mar	surecom	Notification	Down, Unreachable
platao	surecom	Notification	Down, Unreachable
surecomwireless	surecom	Notification	Down, Unreachable

Host Groups

Group Name	Description	Default Contact Groups	Host Members
oriental-servers	ORIENTAL UNDER SERVERS	oriental-admins	oriental , surecom , mar , 241s1
platao-servers	PLATAO Server	platao-admins	platao
surecomwireless-servers	SURECOM WIRELESS UNDER SERVERS	surecomwireless-admins	surecomwireless

Contact Groups

Group Name	Description	Contact Members
oriental-admins	Oriental Nagios Administrators	davidmar
platao-admins	PLATAO Administrators	vmac , davidmar
surecomwireless-admins	SURECOM WIRELESS Administrators	vmac , davidmar

Time Periods

Name	Alias/Description	Sunday Time Ranges	Monday Time Ranges	Tuesday Time Ranges	Wednesday Time Ranges	Thursday Time Ranges	Friday Time Ranges	Saturday Time Ranges
24x7	24 Hours A Day, 7 Days A Week	00:00:00 - 24:00:00	00:00:00 - 24:00:00	00:00:00 - 24:00:00	00:00:00 - 24:00:00	00:00:00 - 24:00:00	00:00:00 - 24:00:00	00:00:00 - 24:00:00
none	No Time Is A Good Time							
nonworkhours	Non-Work Hours	00:00:00 - 24:00:00	17:00:00 - 24:00:00, 00:00:00 - 09:00:00	17:00:00 - 24:00:00, 00:00:00 - 09:00:00	17:00:00 - 24:00:00, 00:00:00 - 09:00:00	17:00:00 - 24:00:00, 00:00:00 - 09:00:00	17:00:00 - 24:00:00, 00:00:00 - 09:00:00	00:00:00 - 24:00:00
workhours	"Normal" Working Hours		09:00:00 - 17:00:00	09:00:00 - 17:00:00	09:00:00 - 17:00:00	09:00:00 - 17:00:00	09:00:00 - 17:00:00	

Commands

Command Name	Command Line
check-host-alive	$USER1$/check_ping -H $HOSTADDRESS$ -w 3000.0,80% -c 5000.0,100% -p 1
check_dns	$USER1$/check_dns -H www.yahoo.com -s $HOSTADDRESS$
check_ftp	$USER1$/check_ftp -H $HOSTADDRESS$
check_hpjd	$USER1$/check_hpjd -H $HOSTADDRESS$ -C public
check_http	$USER1$/check_http -H $HOSTADDRESS$
check_local_disk	$USER1$/check_disk -w $ARG1$ -c $ARG2$ -p $ARG3$
check_local_load	$USER1$/check_load -w $ARG1$ -c $ARG2$
check_local_procs	$USER1$/check_procs -w $ARG1$ -c $ARG2$ -s $ARG3$
check_local_users	$USER1$/check_users -w $ARG1$ -c $ARG2$
check_nntp	$USER1$/check_nntp -H $HOSTADDRESS$
check_ping	$USER1$/check_ping -H $HOSTADDRESS$ -w $ARG1$ -c $ARG2$ -p 5
check_pop	$USER1$/check_pop -H $HOSTADDRESS$
check_smtp	$USER1$/check_smtp -H $HOSTADDRESS$
check_ssh	$USER1$/check_tcp -H $HOSTADDRESS$ -p 22
check_surecom_http	$USER1$/check_tcp -H $HOSTADDRESS$ -p 12345
check_tcp	$USER1$/check_tcp -H $HOSTADDRESS$ -p $ARG1$
check_telnet	$USER1$/check_tcp -H $HOSTADDRESS$ -p 23
check_udp	$USER1$/check_udp -H $HOSTADDRESS$ -p $ARG1$
host-notify-by-email	/usr/bin/printf "%b" "***** Nagios 1.0 *****\n\nNotification Type: $NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState: $HOSTSTATE$\nAddress: $HOSTADDRESS$\nInfo: $OUTPUT$\n\nDate/Time: $DATETIME$\n" | /usr/bin/mail -s "Host $HOSTSTATE$ alert for $HOSTNAME$!" $CONTACTEMAIL$
host-notify-by-epager	/usr/bin/printf "%b" "Host '$HOSTALIAS$' is $HOSTSTATE$\nInfo: $OUTPUT$\nTime: $DATETIME$" | /usr/bin/mail -s "$NOTIFICATIONTYPE$ alert - Host $HOSTNAME$ is $HOSTSTATE$" $CONTACTPAGER$
notify-by-email	/usr/bin/printf "%b" "***** Nagios 1.0 *****\n\nNotification Type: $NOTIFICATIONTYPE$\n\nService: $SERVICEDESC$\nHost: $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState: $SERVICESTATE$\n\nDate/Time: $DATETIME$\n\nAdditional Info:\n\n$OUTPUT$" | /usr/bin/mail -s "** $NOTIFICATIONTYPE$ alert - $HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$ **" $CONTACTEMAIL$
notify-by-epager	/usr/bin/printf "%b" "Service: $SERVICEDESC$\nHost: $HOSTNAME$\nAddress: $HOSTADDRESS$\nState: $SERVICESTATE$\nInfo: $OUTPUT$\nDate: $DATETIME$" | /usr/bin/mail -s "$NOTIFICATIONTYPE$: $HOSTALIAS$/$SERVICEDESC$ is $SERVICESTATE$" $CONTACTPAGER$
process-host-perfdata	/usr/bin/printf "%b" "$LASTCHECK$\t$HOSTNAME$\t$HOSTSTATE$\t$HOSTATTEMPT$\t$STATETYPE$\t$EXECUTIONTIME$\t$OUTPUT$\t$PERFDATA$" >> /var/spool/nagios/host-perfdata.out
process-service-perfdata	/usr/bin/printf "%b" "$LASTCHECK$\t$HOSTNAME$\t$SERVICEDESC$\t$SERVICESTATE$\t$SERVICEATTEMPT$\t$STATETYPE$\t$EXECUTIONTIME$\t$LATENCY$\t$OUTPU

```
T$\t$PERFDATA$" >> /var/spool/nagios/service-perfdata.out
```

Autores

Ricardo David Martins e ***Rui Diogo Lopes***, estudantes do 4º ano de Engenharia de Telecomunicações e Informática do ISCTE, Portugal. Este estudo foi elaborado no âmbito da cadeira de *Inteligência e Gestão em Redes e Serviços*, orientada por Pr. Carlos Sá da Costa.