Administración centralizada de servidores perimetrales.

Memoria Final de Proyecto

TFC – Plataforma GNU/Linux

Ignacio Sergio Pena Filgueira
10 Enero 2011

Índice de contenido.

Índice de contenido. 2
Introducción 5
Finalidad 5
Objetivos 6
Gestión centralizada de los servidores. 6
Estandarización de configuración en todos los clientes. 7
Estandarización de la documentación en todos los clientes. 7
Monitorización 8
Diseño de un protocolo de endurecimiento de servidor. 9
Diseño de plantillas para distintos roles de servidores. 9
Motivación 9
Alcance 12
Infraestructura de distribución de configuraciones. 12
Infraestructura de distribución de paquetes. 12
Infraestructura de monitorización. 12
Instalación del sistema base CentOS 12
Optimización del sistema base. 12
Hardening del sistema base. 13
Plantilla de servidor AntiSpam 13
Plantilla de servidor VPN Roadwarrior 13
Infraestructura de distribución de configuraciones 14
Instalación y configuración. 14
Control de versiones 15
Infraestructura de distribución de paquetes 17
Infraestructura de monitorización. 18
Instalación y configuración Nagios 18
Instalación del sistema base 20
Instalación sistema operativo. 20
Instalación del Agente Puppet 21
Intercambio y firma de certificados. 22
Optimización del sistema base 24
Módulo tunning 24
Clase tunning::ntp 25
Clase tunning::nrpe 26
Clase tunning::munin 27
Clase tunning::utils 29
Clase tunning::cron 30
Clase tunning::repositories 30
Clase tunning::disable_services 30
Hardening de sistema base 32
Módulo de seguridad local 32
Clase local_security::password 32
Clase local_security::defaultrunlevel 33
Clase local_security::sysinit 33
Clase local_security::term 33

Clase local_security::extstorage 34
Clase local_security::fsmount 34
Clase local_security::issue 35
Clase local_security::cronrestrictions 36
Clase local_security::ctrlaltdel 36
Módulo se seguridad de red 37
Clase network_security::sysctl 37
Class network_security::portsentry 38
Clase network_security::ssh 40
Clase network_security::iptables 41
Módulo de comprobación de integridad 42
Clase integrity_check::aide 42
Clase integrity_check::rkhunter 43
Clase integrity_check::lynis 43
Plantilla de servidor AntiSpam 45
Postfix 46
Clase aspam::postfix 46
MailScanner 47
Clase aspam::ms 47
ClamAV 48
Clase aspam::clamd 48
SpamAssassin 50
Class aspam::spamassassin 50
Prueba de funcionamiento antispam y antivirus 50
Plantilla de servidor VPN Roadwarrior 53
Conclusiones 57
Bibliografía 59
Anexos 60
Infraestructura de distribución de configuraciones 60
Infraestructura de monitorización. 60
Instalación del sistema base 63
Optimización del Sistema Base 64
Módulo tunning 64
Clase tunning::ntp 64
Clase tunning::nrpe 65
Clase tunning::munin 66
Clase tunning::utils 66
Clase tunning::cron 67
Clase tunning::repositories 67
Clase tunning::disable_services 67
Hardening del sistema base 69
Módulo se seguridad local 69
Clase local_security::password 69
Clase local_security::defaultrunlevel 70
Clase local_security::sysinit 70
Clase local_security::term 70
Clase local_security::extstorage 71
Clase local_security::fsmount 71

Clase local_security::issue 72
Clase local_security::cronrestrictions 72
Clase local_security::ctrlaltdel 73
Módulo de seguridad de red 74
Clase network_security::sysctl 74
Clase network_security::portsentry 75
Clase network_security::ssh 78
Clase network_security::iptables 79
Módulo de comprobación de integridad 79
Clase integrity_check::aide 80
Clase integrity_check::rkhunter 81
Clase integrity_check::lynis 82
Plantilla de servidor AntiSpam 84
Clase aspam::postfix 84
Clase aspam::ms 85
Clase aspam::clamd 92
Clase aspam::spamassassin 93
Plantilla de servidor VPN Roadwarrior 94

Introducción

Finalidad

Se pretende diseñar e implementar la infraestructura necesaria para implantar, administrar remotamente y monitorizar servidores perimetrales Linux diseñados para estar ubicados en zona desmilitarizada.

La idea a desenvolver es la de crear un producto paquetizado y estable de software y servicio. Como producto paquetizado se entiende una colección de distintos componentes de software optimizada para realizar una determinada tarea, el resultado del trabajo ha de fructificar en además de en estandarización y unificación de configuraciones y documentación, en una simplicidad para el administrador del sistema a la hora de gestionar dichos servidores. Estas tareas de administración y monitorización remotas es el servicio que se venderá como valor añadido a los clientes, frente a una instalación básica de un sistema operativo Linux.

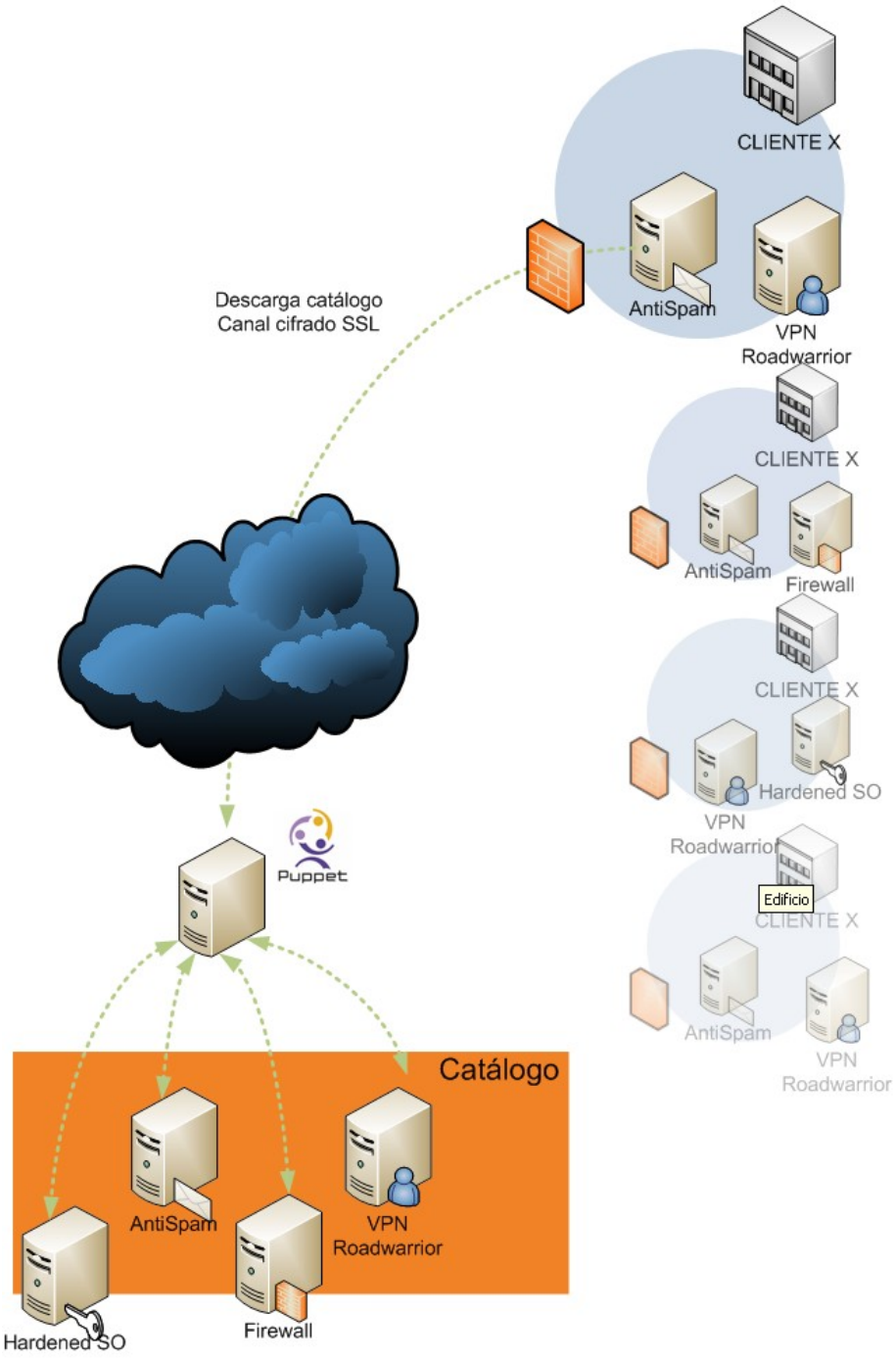
El hecho de realizar un diseño orientado en esta primera fase hacia una zona desmilitarizada frontal, obliga a tomar precauciones especiales en cuanto a la configuración de seguridad. Se busca obtener una instalación base de sistema operativo en el que tras aplicar una serie de buenas prácticas de seguridad, se implementarán medidas de seguridad proactivas y reactivas tanto a nivel de host como de red. Además esta instalación base contará con sistema de monitorización preinstalado, argumento de peso a la hora de plantear un contrato de outsorcing de estos servidores.

La columna vertebral de toda la infraestructura será la implementación de una consola de administración centralizada desde la que se gestionen todos los servidores y en la que se obtenga información del estado de monitorización.

Objetivos

Gestión centralizada de los servidores.

La gestión centralizada de servicios de los servidores, permite disponer de una visión global del conjunto de nodos a administrar, los paquetes de software que se encuentran instalados y las configuraciones utilizadas. Ofrece también la posibilidad de efectuar actualizaciones masivas y un entorno de preproducción para testear las mismas.



Estandarización de configuración en todos los clientes.

Buscamos obtener una base de configuración probada y estable con la que ejecutar todos los servicios en las distintas infraestructuras en casa del cliente. Esto simplificará el mantenimiento debido a la portabilidad de las configuraciones, se realiza una configuración base y sobre ella se personaliza para adaptarla a las necesidades específicas del cliente.

La estructura planteada simplifica también la creación y testeo de prototipos antes de ponerlos en producción. Siempre se trabaja sobre una plantilla, y sobre esta se pueden testear las mejoras correctivas o perfectivas antes de liberar la release correspondiente.

Esta estandarización de configuraciones debilita también la dependencia hacia el administrador de sistemas, ya que administrar diversas infraestructuras heterogéneas en cuanto a configuraciones y paquetes supone una carga de trabajo excesiva y resulta poco productivo.

Además se implementa un control de versiones de configuraciones con lo que se simplifica el seguimiento de mantenimientos evolutivos y rollbacks en caso de fallas producidas por errores de configuración.

Estandarización de la documentación en todos los clientes.

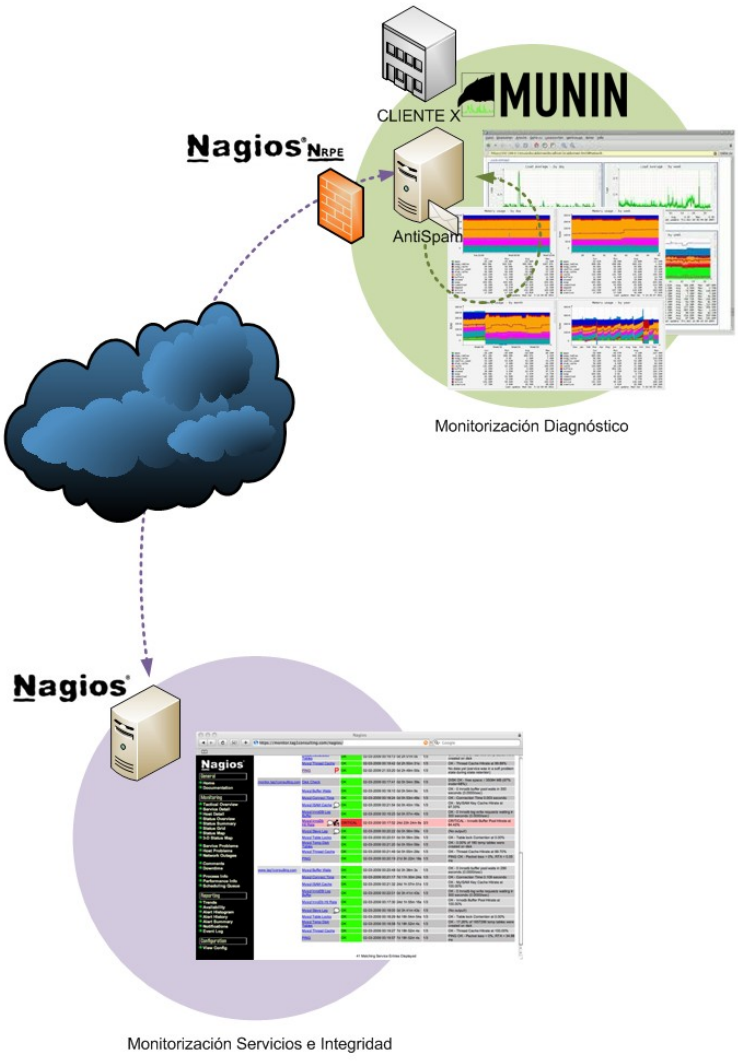
Al disponer de la misma documentación de plantilla para cada uno de los servicios, ésta se ha de realizar sólo una vez, y será posible distribuirla entre todos los clientes tan sólo adjuntando un anexo de personalización. Este anexo se ha de intentar que tenga el menor contenido imprescindible.

Monitorización

Paralelamente a la red de administración y como parte importante del paquete final ofertado, se incluye una monitorización remota de la máquina en cuestión. Se ofertará al cliente opción de adjuntarla a un departamento de operaciones con los correspondientes protocolos de escalado y notificación de los diferentes tipos de alertas detectadas.

A la hora de realizar mantenimientos correctivos o perfectivos en los servidores es preciso disponer de un volumen mayor de información y con una mayor granularidad, que el requerido para la monitorización de servicio.

Se implementará por tanto un segundo sistema de monitorización paralela de diagnóstico. Para no sobrecargar la máquina, esta monitorización se activará bajo demanda y sólo durante el tiempo imprescindible para realizar el diagnóstico.



Diseño de un protocolo de endurecimiento de servidor.

La ventaja de la instalación utilizando plantillas es que se pueden reproducir instalaciones y configuraciones de paquetes completamente automatizadas, por lo cual el grueso del trabajo ha de centrarse en desarrollar estás plantillas que servirán como base para instalar todos los servidores ubicados en las DMZ de los diferentes clientes.

Como parte de la responsabilidad contraída con el cliente a la hora de desarrollar un proyecto se destaca la de elegir la mejor solución disponible independientemente del producto o sistema operativo que la implemente.

La experiencia personal indica que para servidores perimetrales la opción más deseable es la del Software Libre, por motivos que escapan del ámbito de este documento.

Partiendo de que en perimetral es donde Linux ofrece algunas de sus mejores características, y que este perfil de servidor se encuentra expuesto a la mayoría de ataques de Internet, ha de primar la seguridad a la hora de añadir mejoras a la instalación base de del sistema operativo.

Diseño de plantillas para distintos roles de servidores.

Partiendo de esta primera base endurecida se realizarán distintos perfiles de servidor atendiendo a los roles que se pueden llevar a cabo dentro de una DMZ.

- Servidor AntiSpam
- Servidor OpenVPN
- Servidor Firewall

Esta es una línea abierta de trabajo, ya que se pueden continuar desarrollando plantillas según las necesidades detectadas (Servidor WEB, Servidor Proxy, CA, etc)

Motivación

La necesidad de desarrollar esta configuración de infraestructura y de automatizar las tareas descritas anteriormente responde a un crecimiento desordenado de una línea de negocio emergente en una empresa de consultoría TIC.

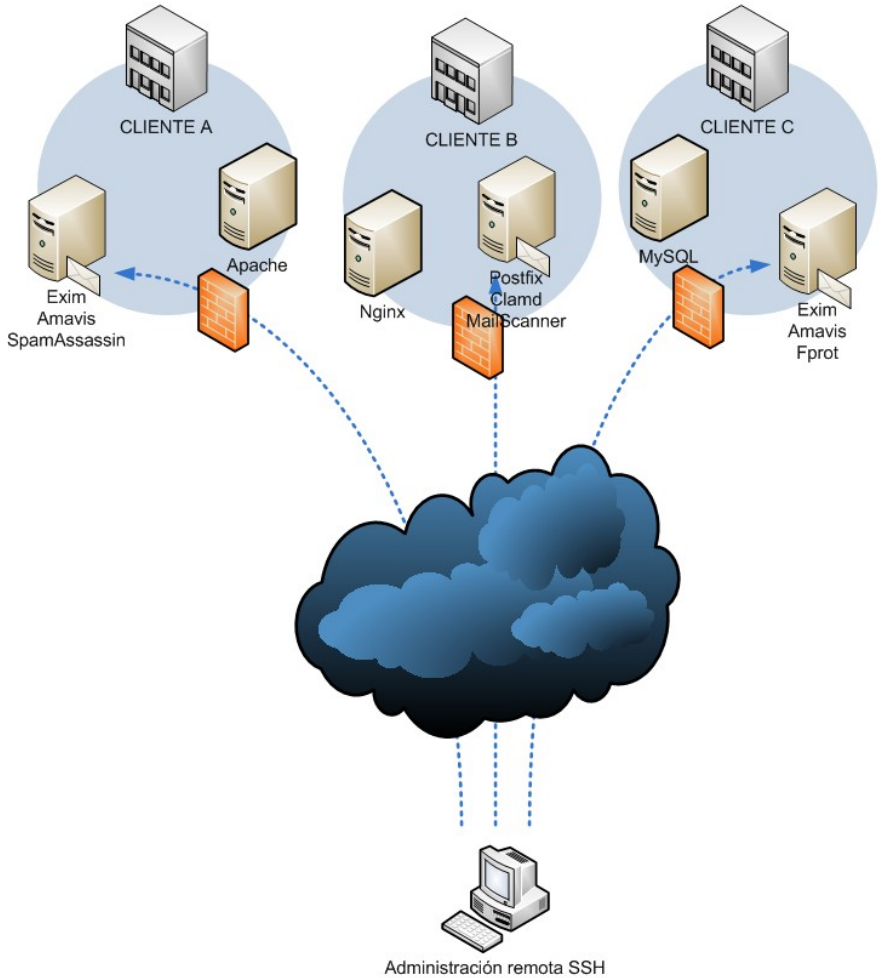
El principal volumen de negocio de esta empresa se centra en PYME y gran empresa, a la que tras asesorar en las necesidades de un servidor perimetral en concreto

y realizar la instalación del mismo, se establecen unos contratos para realizar mantenimiento.

Posteriormente se oferta en una segunda línea un servicio de outsorcing TIC asumiendo la resolución de incidencias tanto en servidores como en puestos de usuario. Esta propuesta vendría a estabilizar esta estructura de negocio, ofreciendo al cliente además de una configuración de sistema operativo estable y segura, la posibilidad de que su servidor esté integrado en la red de administración centralizada, que ofrece como ventajas:

- Seguimiento de actualizaciones.
- Backups de configuración.
- DRP[1].
- Red de monitorización.

Actualmente ya existen contrato de mantenimiento para estos perfiles de servidores, el problema es que esas instalaciones se han ido realizando según iban apareciendo las oportunidades de realizar un proyecto. Cada instalación lleva el sello del administrador que la realizó, personalización de configuraciones, documentación.



1 Realmente reinstalar una máquina que haya caído con la configuración que estaba en ejecución, sería cuestión de dos horas máximo y estaría automatizado

A medida que ha ido creciendo la cartera de clientes con este tipo de contrato, las tareas de administración y su complejidad ha aumentado exponencialmente.

El mantenimiento actual se realiza mediante acceso por consola remota SSH a cada uno de los servidores.

Los distintos servidores perimetrales de los clientes han sido instalados por diferentes personas que ya no se encuentran en la organización y pese a que están debidamente documentados, asumir las tareas de mantenimiento supone un consumo de recursos excesivo, ya que es necesaria una tarea previa de revisión de la documentación del servidor, antes de realizar una intervención en cliente.

Desglosando los inconvenientes de esta situación en las diferentes tareas de mantenimiento, nos encontramos con que todos los mantenimientos correctivos se han de realizar individualmente para cada instalación. Mientras que con una estandarización de software y versiones se podrían realizar los mantenimientos en lotes. Incluso se podría disponer de una instalación piloto sobre la que trabajar, y una vez lista hacer una puesta en producción de la misma, garantizando así la continuidad del servicio.

Estas mismas dificultades también lastran cualquier tipo de mantenimiento perfectivo sobre la infraestructura, convirtiéndose en algo flagrante cuando hay que actualizar algún paquete de software debido a la necesidad de nuevas funcionalidades o a vulnerabilidades detectadas y explotables.

El mantenimiento evolutivo también se encuentra negativamente afectado por esta heterogeneidad de sistemas, cualquier nueva instalación de software requiere de un nuevo análisis de compatibilidad de versiones y librerías.

Así pues la motivación para desarrollar este proyecto se centra en reducir los recursos necesarios para el mantenimiento y definir unas directrices futuras para las nuevas implantaciones en futuros clientes. La principal ventaja es la simplificación de las tareas de mantenimiento, ya que se gana una visión global de todas las infraestructuras otorgando un mayor control sobre cualquier aspecto de las mismas. El punto fuerte de la instalación es la interfaz de administración centralizada, ya que solventa todos los inconvenientes anteriormente descritos y reduce los tiempos de planificación e intervención drásticamente.

Se pretende ganar también un sólido argumento de venta, ya que en la labor comercial no se ofrece la instalación de un sistema operativo Linux, si no que se ofrece un producto paquetizado con un plan de mantenimiento, auditoría, monitorización y seguimiento asociado.

Alcance

Debido a la naturaleza del proyecto se ha de acometer la instalación de varias infraestructuras paralelamente, distribución de paquetes, configuraciones y monitorización. Por otro lado se ha de realizar un proceso de instalación y endurecimiento de un sistema CentOS 6 y modelar este perfil en plantillas de configuración compatibles con la herramienta de distribución. También se ha de realizar el mismo proceso con una instalación de cada tipo de perfil de servidor a distribuir.

Infraestructura de distribución de configuraciones.

Se utilizará para este cometido Puppet, se trata de una herramienta de automatización de configuraciones de servidores. Utiliza una arquitectura cliente-servidor, por tanto se ha de detallar la instalación y configuración del servidor y procedimentar la instalación y configuración del cliente.

Infraestructura de distribución de paquetes.

Durante el desarrollo del proyecto surgió la necesidad de contar con un repositorio propio de paquetes. Se detalla como instalar y configurar un repositorio[2] accesible por yum.

Infraestructura de monitorización.

Para la monitorización de servicios y alertas se utilizará una sonda Nagios[3][4] que realizará comprobaciones activas utilizando el paquete NRPE instalado sobre los clientes. Se ha de realizar por tanto la instalación de una sonda Nagios a la que conectaremos todos los clientes instalados mediante puppet.

La instalación final también incorporará una segunda monitorización a cargo de Munin, se utilizará para tareas de diagnóstico por lo que se define como una monitorización sin emisión de alertas y con una mayor granularidad y detalle.

Instalación del sistema base CentOS

Se detallan las guías seguidas durante la instalación del sistema operativo.

Optimización del sistema base.

Se realizan tareas de tuning, incorporación de herramientas y servicios útiles para realizar la administración del servidor. Instalación del servicio de sincronización

2 Mantenimiento evolutivo: SSL en servicio httpd del repositorio.
3 Mantenimiento evolutivo: Instalación plugins de Nagios, pmp4nagios, bussiness intelligente...
4 Mantenimiento evolutivo: Conexión de sonda Nagios a base de datos con NDO

horaria, planificador de tareas, repositorios adicionales, eliminación de servicios innecesarios e instalación del agente de monitorización. De esta optimización del sistema han de salir los manifiestos de Puppet que automaticen la realización de estas configuraciones.

Hardening del sistema base.

Como hardening[5] entendemos unas medidas aplicables a un servidor con el fin de minimizar el riesgo de intrusión en el mismo. Se incluyen aquí medidas de seguridad local, de red, instalación de herramientas de seguridad proactiva y de integridad de datos. También se ha de conectar a la sonda de monitorización aquellas herramientas cuya salida tenga que ser supervisada. De esta etapa se obtendrán tres módulos del sistema de manejo de configuraciones, dos para establecer las medidas de seguridad local y de red, y un tercero para la comprobación de integridad de datos.

Plantilla de servidor AntiSpam

Se plantea aquí el diseño de un servidor AntiSpam perimetral, se presupone que las funciones de MDA las realizará otro servidor en otra DMZ menos restrictiva, por ejemplo CyrusIMAP o Microsoft Exchange.

Se confía para realizar las funciones de agente de transporte de correo en Postfix[6], las funciones de seguridad de correo las realizará MailScanner[7], como antivirus se configurará una instancia de ClamAV en modo servicio, y para la búsqueda y puntuación de correo no deseado se instalará una instancia de SpamAssassin. La configuración de estos paquetes se realizará mediante un módulo de Puppet para cada uno de ellos.[8][9]

Plantilla de servidor VPN Roadwarrior

Se instalará en este módulo, una instancia de OpenVPN para el acceso a la red local desde internet.

5 Mantenimiento evolutivo: Establecimiento de servicios de logs centralizados, tan sólo hay que configurar el rsyslog que viene por defecto en CentOS 6
6 Mantenimiento evolutivo: Enjaular los demonios de Postfix
7 Mantenimiento evolutivo: Montar tmpfs para la descompresión de archivos adjuntos
8 Mantenimiento evolutivo: Instalar pyzor.
9 Mantenimiento evolutivo: Implementar autenticación SPF.

Infraestructura de distribución de configuraciones

Como ya se ha mencionado anteriormente para la distribución de políticas de configuración a los clientes se utilizará Puppet, una herramienta escrita en Ruby y liberada actualmente bajo licencia Apache 2.0. La potencia de este software radica en que dispone de un lenguaje declarativo, en el que podemos definir entidades y relaciones a nuestro antojo. Utiliza una arquitectura cliente-servidor, el servidor se encarga del mantenimiento, compilación y distribución de catálogos de configuraciones. Además Puppet incorpora un servidor de ficheros propio. El cliente consta de un demonio que descarga el catálogo del servidor y se encarga de implementarlo en la máquina periódicamente. Esto otorga otra ventaja, es posible definir un estado deseado de la máquina cliente y el agente de Puppet en cada ejecución revertirá cualquier cambio realizado en la misma.

Haremos también uso de Augeas, una utilidad de edición de ficheros de configuración que se integra perfectamente con Puppet. Se encarga de mapear las opciones de un fichero de configuración en una estructura arbórea e integrarla en el esquema del sistema de archivos de la máquina. Para hacer esto Augeas[10] crea “lentes”, ficheros de patrones de expresiones regulares específicos que definen los parámetros para un tipo de fichero de configuración concreto. Es por esto por lo que no podremos editar todos los archivos de configuración que nos encontremos sin entrar a desarrollar módulos para esta herramienta.

Puppet utiliza una base de datos sqlite3 para almacenar la información y catálogos, es recomendable para sistemas en producción conectar el demonio a una base de datos MySQL. Para la puesta en producción también es muy recomendable instalar Mongrel con un proxy para las conexiones SSL, esto es porque como servidor web la instalación por defecto utiliza WEBrick, un servidor sencillo y no escalable.

Instalación y configuración.

El paquete de software se encuentra disponible para CentOS en los repositorios EPEL[11], podremos instalarlos ejecutando el siguiente comando:

```
#rpm -Uvh http://download.fedora.redhat.com/pub/epel/6/i386/epel-
release-6-5.noarch.rpm
```

Se ha de realizar la instalación de los paquetes puppet-server y Augeas utilizando la herramienta yum.

```
#yum install puppet-server augeas
```

10 Línea Abierta de estudio: Desarrollar lentes para Augeas.
11 Extended Packages Enterprise Linux

Tras instalar el agente Puppet y añadirlo a la configuración de arranque con `chkconfig`, también se ha de ha de ejecutar el siguiente comando para crear los usuarios y asignar los permisos adecuados con los que correrá el demonio, para ello ejecutar:

```
#puppet agent –mkusers
```

Tras ejecutar este comando se pueden arrancar el servicio con los scripts propios de la distribución. Se configura en el arranque con chkconfig.

```
#chkconfig --add --level 345 puppet-master
#chkconfig puppet-master on
```

En el servidor hemos de abrir el puerto en el que escucha el servidor de Puppet, añadimos la siguiente regla al archivo de iptables.

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 8140 -j ACCEPT
```

Tras esto se ha de reiniciar iptables para aplicar los cambios.

```
#service iptables restart
```

Ficheros anexos
/etc/puppet/puppet.conf

Control de versiones

Una buena práctica recomendada en las instalaciones de infraestructura de Puppet, es la instalación de un repositorio de versiones para llevar un control exhaustivo de los cambios realizados en los ficheros de módulos. De esta forma se lleva un registro de cambios, historial de versiones de configuración y así se puede realizar rollback a cualquier configuración anterior.

Se debe aplicar un control de versiones tanto a los manifiestos de Puppet como a los ficheros de configuración servidos a los clientes.

Como herramienta de control de versiones se utilizará Bazaar, el motivo para elegir la herramienta de Canonical frente a otras más utilizadas como SVN o CVS, es la sencillez de configuración y la simplicidad de uso para instalaciones locales.

```
#yum install bzr
```

Tras esto posicionarse en el directorio /etc/puppet e indicarle a Bazaar que inicie un repositorio en este directorio con el comando:

```
#bzr init
Created a standalone tree (format: 2a)
```

Se añadirán al repositorio los ficheros de los que se desea mantener control de versiones con el siguiente comando.

```
# bzr add modules/*
adding modules
adding modules/ntp
adding modules/ssh
adding modules/ntp/files
adding modules/ntp/manifests
...
```

Ya está iniciado el control de versiones para estos ficheros, se han de añadir al repositorio los nuevos ficheros creados, y recordar realizar commit para salvar las modificaciones realizadas. Se ha de documentar cada cambio en los ficheros de configuración y scripts de los módulos al realizar commit en el repositorio de Bazaar.

Se utilizarán los siguientes comandos para añadir y actualizar ficheros en las revisiones.

- bzr add file → Añade fichero al repositorio.
- bzr commit file → Salva los cambios y solicita comentario

Infraestructura de distribución de paquetes

Se hace presente durante el desarrollo del proyecto la necesidad de disponer de un repositorio propio de paquetes desde el que se puedan distribuir rpms creados a partir de código fuente. No es reproducible desde puppet configurar, compilar e instalar software del que no se dispone un paquete rpm. Si bien puppet ofrece un servidor de ficheros integrado desde el que se pensó en un primer momento distribuir los rpm's creados, no es factible ya que aunque el rpm está accesible desde la máquina, ese origen de paquetes no está soportado por yum.

La creación de un repositorio es una tarea sencilla si se dispone de un servidor web ya configurado, tan sólo hemos tenido que mover los paquetes a una carpeta de este servidor web y ejecutar el comando createrepo, proporcionado por un paquete instalable por yum con el mismo nombre.

```
#yum install createrepo
```

Una vez hemos instalado el paquete y situado los rpms en la ruta del servidor web tan solo hemos de ejecutar desde la línea de comandos

```
#createrepo /var/www/html/repo
```

Esto creará un directorio en la ruta indicada con los ficheros de índices de paquetes. Tan sólo tendremos que agregar este repositorio a los clientes de puppet mediante un guión que distribuirá el siguiente fichero de configuración y lo posicionará en /etc/yum.repos.d/repo.repo.

```
[repo.repo]
name=Repo Local
baseurl=http://master/repo/
enabled=1
gpgcheck=0
```

Infraestructura de monitorización.

Para la monitorización de los servidores se confiará en Nagios Core, se trata un sistema de monitorización de redes de código abierto, publicado bajo licencia GPLv2 según la FSF. Nagios destaca frente a otros sistemas de monitorización por la gran plasticidad de la que dispone al basarse en un sistema de PlugIns para realizar su trabajo. Es capaz de monitorizar gran cantidad de servicios y tipos distintos de hosts de forma activa, pasiva o mixta. También ofrece una gran variedad de sistemas de notificación, y dispone de grupos de escalado en caso de no reconocimiento de la alarma.

Debido a que utilizaremos comprobaciones activas sobre los nodos a monitorizar, además de la instalación de sonda deberemos instalar en cada nodo el conector que nos permita ejecutar los scripts de monitorización. Entre las varias alternativas disponibles utilizaremos NRPE, el paquete más común a la hora de realizar comprobaciones activas de monitorización.

Instalación y configuración Nagios

Para instalar la sonda de Nagios únicamente deberemos instalar el paquete Nagios, disponible desde los repositorios EPEL. Como dependencia se instalará Apache para servir el cgi de la interfaz web. La configuración de Nagios se realiza en el fichero /etc/nagios/nagios.cfg.

Si existe cualquier error de configuración en los ficheros que definen los objetos de Nagios, bien sean hosts, services, contacts, etc, el demonio no arrancará por lo que es muy importante realizar un chequeo para depurar estos mensajes de error.

Lanzando el binario con el flag -v (verify) seguido del fichero de configuración se realizará una comprobación de la consistencia de estas definiciones y nos aseguraremos de que el demonio arrancará correctamente.

El único paso necesario para acceder a la interfaz web es crear el fichero de password con el mecanismo de autenticación de Apache. En el fichero de configuración del site apache /etc/httpd/conf.c/nagios.cfg podemos ver la ruta en la que se espera este fichero que hemos de generar con el comando htpasswd.

```
AuthUserFile /etc/nagios/passwd
```

Mediante el comando htpasswd crearemos un usuario y password para acceder a la interfaz de Nagios

```
#htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
New password:
Re-type new password:
Adding password for user nagiosadmin
```

Una vez está realizada esta configuración básica de nagios hemos de instalar el plugin que utilizaremos para realizar las comprobaciones remotas, instalaremos el paquete nagios-plugins mediante yum. Este paquete contiene el binario check_nrpe que es el que lanzará las comprobaciones remotas en los clientes.

En las reglas de firewall hemos de permitir el tráfico entrante para el puerto 80 y permitir tráfico saliente hacia el puerto 5556, puerto de destino en el que estará escuchando el demonio NRPE.

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A OUTPUT -m state --state NEW -m tcp -p tcp --dport 5666 -j ACCEPT
```

Tras esto se ha de reiniciar iptables para aplicar los cambios.

```
#service iptables restart
```

Tras esto si accedemos a la URL http://[IP_SERVIDOR/nagios veremos ya disponible la interfaz web de monitorización.

Ficheros Anexos
/etc/nagios/nagios.cfg

Instalación del sistema base

Instalación sistema operativo.

Se realizará una instalación limpia del sistema operativo CentOS 6, tan sólo se atenderá a un par de puntualizaciones a la hora de realizar el particionado de disco. Anaconda, el instalador de CentOS/RedHat dispone de interfaz gráfica y modo texto con ncurses. En el hemos de configurar las opciones por defecto como distribución de teclado, zona horaria, etc.

A la hora de particionar el disco se dispondrá de una partición primaria en la que se montará /boot. El resto de particiones se definirán sobre LVM2, gestor de volúmenes que nos ofrecerá mayor flexibilidad a la hora de dimensionar volúmenes, añadir mayor capacidad mediante nuevos volúmenes físicos o realizar snapshots en caliente de los sistemas de ficheros. El motivo de no incluir boot en un volumen es que Grub no sería capaz de acceder a este tipo de particionado, por lo que si lo configurásemos de este modo nos encontraríamos con un sistema que no arranca ya que no se encuentra kernel, initrd, system-map, etc.

Así pues destinamos 200 Mb para /boot y el resto lo configuraremos como un grupo de volúmenes.

La distribución de los volúmenes se realizará de acuerdo a las guías reseñadas en el FHS[12], documento que estandariza los sistemas de fichero *nix. En el se especifica que partes del sistema de ficheros son susceptibles de crecer en volumen, y cuales es posible compartir entre varias máquinas mediante NFS. Nos interesará separar tres directorios en distintos volúmenes.

- home, en este directorio se ubican todos los archivos personales de los usuarios del sistema (excepto root),
- var, aquí se ubican ficheros logs, librerías, locks, buzones de correo, etc. Cualquier fichero de tamaño variable.
- tmp directorio de ficheros temporales del sistema.

Para evitar que el crecimiento de algún fichero de usuario ocupe todo el sistema de ficheros raíz de la máquina y produzca una denegación de servicio y un estado inconsistente, aislaremos estos directorios en volúmenes aparte. Además esto nos permitirá definir opciones de montaje específicas que nos interesarán por motivos de seguridad.

Una vez realizada la instalación Anaconda deja un fichero en /root llamado Anaconda-ks.cfg que contiene los parámetros de configuración especificados durante la instalación, mediante kickstart podremos replicar esta configuración en instalaciones

12 Filesystem Hierarchy Standard

desatentidas pasando el parámetro “ks=http://[ruta_servidor]/[fichero-ks]” al kernel durante el proceso de arranque del instalador.

Ficheros Anexos
/var/www/html/ks.cfg

Instalación del Agente Puppet

Una vez arrancado el sistema deberemos instalar los repositorios extendidos mediante el comando:

```
rpm -Uvh http://download.fedora.redhat.com/pub/epel/6/i386/epel-release-6-5.noarch.rpm
```

Una vez instalado el repositorio descargaremos y configuraremos el agente de puppet para que busque los catálogos de configuración en el servidor. También realizaremos el intercambio y firma de certificados SSL.

Instalamos el paquete puppet mediante el gestor de paquetes yum

```
#yum install puppet
```

Tras instalar el agente lo añadiremos a la configuración de arranque para que se ejecute automáticamente en los runlevels 3, 4 y 5.

```
#chkconfig --add --level 345 puppet
#chkconfig puppet on
```

También se ha de ejecutar el siguiente comando para crear los usuarios y asignar los permisos adecuados con los que correrá el demonio:

```
#puppet agent --mkusers
```

Si no ejecutásemos el comando anterior veríamos una línea en /var/log/messages indicando que no se puede obtener el certificado para ese usuario, además de problemas para crear los directorios de la CA, ya que este script se encarga de crear estos usuarios en el sistema.

```
Nov 18 18:15:23 master puppet-master[2099]: Could not create resources
for managing Puppet's files and directories in sections [:main, :ssl,
:ca]: Could not find a default provider for user
```

En el fichero de configuración se ha de especificar la dirección del servidor maestro, mediante la opción de configuración server dentro de sección [main]

```
server = [FQDN SERVIDOR]
```

Es importante mencionar aquí que al establecer las relaciones de confianza utiliza como CN el FQDN del servidor, por lo que ha de estar correctamente configurados los registros DNS en la red. Así también en los ficheros de configuración nunca se referencia por dirección IP un nodo.

Editamos el fichero /etc/sysconfig/iptables y añadimos la siguiente línea para permitir las conexiones salientes al puerto 8140

```
-A OUTPUT -m state --state NEW -m tcp -p tcp --dport 8140 -j ACCEPT
```

Intercambio y firma de certificados.

Una vez arrancado el servicio se observa que ya se genera la solicitud de firma de certificado en el servidor maestro:

```
Nov 18 18:16:43 master puppet-master[2164]: Could not find
certificate_request for 'centos.test'
Nov 18 18:16:43 master puppet-master[2164]: centos.test has a waiting
certificate request
```

Y el agente indica que no se puede verificar el certificado para esta sesión SSL

```
[root@master ~]#puppet agent --verbose --server master --waitforcert 120
--test
warning: peer certificate won't be verified in this SSL session
warning: peer certificate won't be verified in this SSL session
info: Creating a new SSL certificate request for centos.test
Nov 18 18:16:42 centos puppet-agent[2172]: Creating a new SSL
certificate request for centos.test
info: Certificate Request fingerprint (md5):
36:50:C4:99:EC:73:6A:E3:10:13:8A:75:97:9E:2D:F8
Nov 18 18:16:42 centos puppet-agent[2172]: Certificate Request
fingerprint (md5): 36:50:C4:99:EC:73:6A:E3:10:13:8A:75:97:9E:2D:F8
warning: peer certificate won't be verified in this SSL session
```

La opción de waitforcert intentará obtener una sesión SSL con el certificado firmado por la CA del master cada 120 segundos, con la opción –-verbose se muestran con mayor detalle los mensajes informativos del demonio.

Si en el maestro ahora se solicita la lista de certificados pendientes de firmar veremos que centos.test ya se encuentra disponible.

```
[root@master ~]# puppet cert --list
Nov 18 18:21:23 master puppet-master[2164]: Could not find certificate
for 'centos.test'
  centos.test (36:50:C4:99:EC:73:6A:E3:10:13:8A:75:97:9E:2D:F8)
```

Tras verificar que el hash md5 coincide con la suma de integridad mostrada por el agente podemos proceder a firmar el certificado en el servidor para que de este modo quede establecida la asociación de confianza, por defecto la caducidad de los certificados autogenerados por puppet es de 5 años.

Con el siguiente comando se firma el certificado y se reconoce el títere como nodo a gestionar desde el servidor maestro.

```
[root@master ~]# puppet cert --sign centos.test
notice: Signed certificate request for centos.test
notice: Removing file Puppet::SSL::CertificateRequest centos.test at
'/var/lib/puppet/ssl/ca/requests/centos.test.pem'
```

Tras esto en los logs del cliente vemos que ya estamos asociados con el servidor.

```
Nov 18 18:28:56 centos puppet-agent[2782]: Applying configuration
version '1321896525'
```

Optimización del sistema base

Módulo tunning

A partir de este momento todas las configuraciones e instalaciones de paquetes las definiremos como módulos de puppet. Cada módulo describe una estructura de directorio definida

Puppet irá a buscar esos módulos para compilarlos en los catálogos a la ruta /etc/puppet/modules, dentro de este directorio se tiene que crear otro directorio para el módulo que se está desarrollando con su mismo nombre, y los subdirectorios ./module_name/files y ./module_name/manifests. Se ejecuta para ello el siguiente comando.

```
#mkdir -p /etc/puppet/modules/module_name/{files,manifests}
```

Dentro de /etc/puppet/modules/module_name/manifests se ubicarán las definiciones de clases que se aplicarán a los nodos. Así también dentro de /etc/puppet/modules/module_name/files estarán los ficheros mapeados en el servidor de ficheros.

Así pues para el módulo de optimización crearemos el fichero:

```
/etc/puppet/modules/tuning/manifests/init.pp
# init.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>

# See LICENSE for the full license granted to you.

# Main module class
#

class tunning{
       include tunning::ntp
       include tunning::ntp::nrpe
       include tunning::cron
       include tunning::disable_services
       include tunning::nrpe
       include tunning::repositories
       include tunning::utils
}
```

Dentro de este fichero init se definirán todas las clases que pertenecen al módulo, en este caso definiremos la clase npt que codificaremos a continuación

Ficheros anexos
/etc/puppet/modules/tunning/manifiests/init.pp

Clase tunning::ntp

Se pretende instalar y mantener actualizada la hora del sistema mediante la codificación de una clase en el que se definirán los recursos y ficheros de configuración que se servirán a los agentes, con el fin de realizar una instalación funcionante del cliente ntp.

Es imprescindible mantener la hora del reloj sincronizada, sobre todo para correlación de logs, ya que como mantenimiento evolutivo se recomienda añadir un servidor de logs centralizado a la infraestructura.

```
/etc/puppet/modules/tunning/manifests/ntp.pp
# ntp.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs & configure ntpd
#
class tunning::ntp ($enable = true, $ensure = running) {
       $service_name = 'ntpd'
       $pkg_name = 'ntp'

       package { 'ntp':
              ensure       =>    installed,
       }

       service { 'ntpd':
              ensure =>     $ensure,
              name   =>     $service_name,
              enable       =>    $enable,
              subscribe => File['ntp.conf'],
       }

       file { 'ntp.conf':
              path   =>     '/etc/ntp.conf',
              ensure =>     file,
              require =>    Package['ntp'],
       }
}
```

Sirva esta clase para ejemplificar el lenguaje declarativo que utiliza Puppet, en el podemos ver como se definen recursos y sus dependencias, la mayoría de las configuraciones se basan en el trío paquete-servicio-configuración. En concreto aseguramos que el paquete ntp está instalado, en caso de que no se encuentre se procedería a instalarlo antes de procesar el resto de recursos.

En segundo lugar está el servicio ntpd, cada 30 minutos el agente de puppet se encargará de comprobar que el demonio ntpd está ejecutándose y si no lo encontrase lo lanzaría de nuevo. Además cada vez que se modifique el fichero de configuración ntp.conf puppet se encargará de reiniciar el demonio.

Ahora en el fichero /etc/puppet/manifests/sites es donde definiremos que a este nodo se le aplicará este módulo en concreto, y pasándole los parámetros necesarios:

```
/etc/puppet/manifests/sites
node nodos {
       class {'tunning::ntp':
              ensure => stopped,
       }
}
```

Denifimos también la instalación del comando NRPE y binario de comprobación de nagios para este servicio en concreto.

```
/etc/puppet/modules/tunning/manifests/ntp/nrpe.pp
# nrpe.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

class tunning::ntp::nrpe{

       package{
              'nagios-plugins-ntp':
                     ensure =>     installed,
       }
       file{
              'nrpe.ntp.cfg':
                     ensure =>     file,
                     path   =>     "/etc/nrpe.d/nrpe.ntp.cfg",
                     content      =>
       "command[check_ntp_time]=/usr/lib/nagios/plugins/check_ntp_time
-H 0.pool.ntp.org",
                     notify =>     Service['nrpe'],
       }
}
```

Siempre que se instale una clase y proceda, se instalarán los elementos necesarios para obtener información de monitorización de la misma. En este caso se genera un fichero en el directorio nrpe que incorporará todos los ficheros presentes a su configuración. Definimos en este fichero el comando que se solicita ejecutar de forma remota. La instalación de nrpe se realiza sin la posibilidad de incluir parámetros en los comandos que se solicitan desde la sonda nagios. Esta decisión obliga a definir los comandos estáticamente, pero aumenta la seguridad del demonio.

Ficheros Anexos
/etc/puppet/modules/tunning/manifests/ntp.pp
/etc/puppet/modules/tunning/manifests/ntp/nrpe.pp

Clase tunning::nrpe

Cada instalación básica de sistema se configurará con una instalación de Nagios Remote Plugin Executor, de tal forma que para añadir el nodo a un sistema de configuración tan sólo se tenga que dar de alta este en la sonda Nagios.

Se ha de realizar por tanto la instalación en cliente del paquete nrpe.

La configuración de NRPE permite ejecutarlo como un demonio específico o encapsularlo dentro de xinetd, aportando una capa de seguridad extra al poder especificar desde que direcciones IP se aceptan conexiones contra este servicio.

Dado que los ficheros de configuración si se dispone de una única sonda de monitorización serán los mismos, se especifican una sola vez y serán distribuidos por puppet.

El módulo se encarga de instalar el paquete e instalar los plugins necesarios para realizar las comprobaciones básicas del sistema. Se comprobará ocupación de discos, memoria, swap, número de procesos.

NRPE utiliza autenticación SSL, deberemos incluir una directiva en el fichero de configuración para indicar los nodos desde los que permitirán conectar para obtener la información de las comprobaciones. Este fichero se ubica dentro de /etc/nrpe/conf.d y contiene una línea con la directiva allowed_hosts, para generar este fichero utilizamos plantillas ERB de puppet, lo que nos permitirá definir la dirección de la sonda de nagios en tiempo de ejecución.

Se han de habilitar las conexiones entrantes al puerto 5666/tcp en el servidor.

```
-A INPUT -m state --state NEW -m tcp -p tcp --dport 5666 -j ACCEPT
```

Ficheros Anexos
- /etc/puppet/modules/tunning/manifests/nrpe.pp
- /etc/puppet/modules/tunning/templates/nrpe.server.erb
- /etc/puppet/modules/tunning/files/nrpe.general.commands

Clase tunning::munin

Munin es un sistema de monitorización de host, permite obtener información detallada sobre el rendimiento de la máquina. Permite conectarse a una sonda centralizada al igual que Nagios, y se integra con el pudiendo enviarle alertas en caso de que un valor monitorizado supere un determinado umbral. Enviar tal cantidad de información a través de internet es excesivo por lo que se toma la decisión de dejar este segundo sistema como monitorización de diagnóstico. Se instalan los paquetes de nodo y servidor en la instalación base y se preparan los scripts de Puppet que se encargarán de la instalación del servicio.

Obtendremos por defecto en la instalación de Munin sobre la plantilla de servidor AntiSpam los siguiente parámetros:

- Discos
  - Operaciones de E/S por dispositivo.(IOs/sec)
  - Latencia de disco.(IO wait) (segundos)

    - Throughput por dispositivo.(Bytes/sec)
    - Porcentaje de uso por partición.
    - Porcentaje de uso por dispositivo.
    - Porcentaje de uso de inodos.
    - Latencias de disco por operaciones (R/W)
    - IOstat, blockes por segundo.
- Munin
    - Tiempos de proceso.
        - Actualización.
        - Generación de gráficos.
        - Generación de html
- Red
    - Errores de tarjeta de red (packet/sec)
    - Tráfico de tarjeta de red.
    - Entrada salida de firewall
    - Conexiones activas.
        - Activas
        - Pasivas
        - Fallidas
        - Resets
        - Establecidas
- Postfix
    - Caudal de datos
    - Número de mail encolados.
- Procesos
    - Número de forks por segundo.
    - Número de threads.
    - Número de procesos.
    - Prioridad de procesos en ejecución.
    - Estado de procesos (running vs I/O sleep)
- Sistema
    - Entropía
    - Uso de CPU
        - System
        - User.
        - Nice.
        - Idle
        - iowait
        - IRQ
        - SotfIRQ
        - Steal
    - Número de ficheros abiertos.
    - Interrupciones/sec
    - Número de inodos abiertos.

    - Carga media de CPU
    - Usuarios logados.
    - Uso de memoria
        - Aplicaciones
        - Tablas de páginas.
        - Cache de swap
        - Caché
        - Buffers
        - No usada
        - Swap
        - Inactiva.
        - Activa
        - Mapeada
- SELinux
    - Busquedas AVC
    - Aciertos AVC
    - Errores AVC
- Swap (I/O)
- Uptime

Como vemos genera muchísima información sobre el estado general de la máquina, se genera el script de Puppet para la instalación de este sistema de monitorización.

Ficheros Anexos
/etc/puppet/modules/tunning/manifests/munin.pp

Clase tunning::utils

En esta clase se instalarán paquetes y utilidades frecuentemente usados en la administración de los servidores. Simplemente añadiremos aquí todas esas herramientas que queremos tener disponibles en las instalaciones base.

- htop, sustituto de herramienta gnu top utilizando ncurses. Permite desplazarse por los procesos en ejecución y ofrece más información que la habitual herramienta.
- man, el gestor de páginas de manual se encuentra por defecto en la instalación base de CentOS 6. Se instala en todas las máquinas desde este módulo.
- Tree, herramienta GNU que permite ver una estructura de subdirectorios en forma arbórea.
- CCZE, colorizador de logs. Muy útil a la hora de examinar logs en tiempo real combinándolo mediante un pipe con el comando tail -f.

Se genera el correspondiente módulo de puppet que se encarga de asegurarse que esos paquetes permanecen instalados en el servidor.

Ficheros Anexos
/etc/puppet/modules/tunning/manifests/utils.pp

Clase tunning::cron

En el sistema operativo *nix, cron es un administrador regular de procesos en segundo plano. El demonio se encarga de lanzar procesos o guiones a intervalos regulares (cada minuto, día, semana...)

Por defecto la instalación no viene con ningún programador de tareas instalado, creamos la clase correspondiente para instalar y configurar este servicio.

Este paquete es necesario en el sistema y generará dependencias para multitud de módulos que se utilizarán posteriormente.

Ficheros Anexos
/etc/puppet/modules/tunning/manifests/cron.pp

Clase tunning::repositories

Se codifica una clase exclusivamente para dar de alta los repositorios alternativos y el repositorio base, tan sólo se carga un fichero en el host en el directorio /etc/yum.repos.d con el formato adecuado para que se cargue esta nueva fuente de paquetes en yum.

Ficheros Anexos
/etc/puppet/modules/tunning/manifests/repostories.pp
/etc/puppet/modules/tunning/files/repo.repo

Clase tunning::disable_services

Es una buena práctica deshabilitar todos aquellos servicios que no se van a utilizar del arranque chkconfig, para hacernos una idea podemos ejecutar netstat -lnp para comprobar todos los servicios a la escucha. De esta forma además de mejorar el rendimiento de la máquina, minimizamos la exposición de servicios que pueden ser vulnerables a la red.

```
#netstat -tlnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address
State       PID/Program name
tcp        0      0 0.0.0.0:35343               0.0.0.0:*
LISTEN      1081/rpc.statd
tcp        0      0 0.0.0.0:111                 0.0.0.0:*
LISTEN      1063/rpcbind
tcp        0      0 0.0.0.0:22                  0.0.0.0:*
LISTEN      1179/sshd
tcp        0      0 :::58212                    :::*
LISTEN      1081/rpc.statd
tcp        0      0 :::111                      :::*
LISTEN      1063/rpcbind
```

```
tcp        0      0 :::22                       :::*
LISTEN      1179/sshd
udp        0      0 0.0.0.0:54789               0.0.0.0:*
1081/rpc.statd
udp        0      0 0.0.0.0:814                 0.0.0.0:*
1063/rpcbind
udp        0      0 0.0.0.0:833                 0.0.0.0:*
1081/rpc.statd
udp        0      0 0.0.0.0:111                 0.0.0.0:*
1063/rpcbind
udp        0      0 :::814                      :::*
1063/rpcbind
udp        0      0 :::46521                    :::*
1081/rpc.statd
udp        0      0 :::111                      :::*
1063/rpcbind
```

Tras deshabilitar los servicios NFS y FCOE, podemos ver que sólo se encuentra el demonio de ssh a la escucha.

```
[root@localhost ~]# netstat -lnp
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address               Foreign Address
State       PID/Program name
tcp        0      0 0.0.0.0:22                  0.0.0.0:*
LISTEN      1179/sshd
tcp        0      0 :::22                       :::*
LISTEN      1179/sshd
```

Creamos una clase puppet denominada disable_services que incluiremos en el módulo tunning. Esta clase lo único que utilizará son las capacidades de puppet para manejar servicios asegurándose de que está en estado stopped y de que no están incluidos en el arranque de la máquina.

Ficheros Anexos
/etc/puppet/modules/tunning/manifests/disableservices.pp

Hardening de sistema base

Englobaremos todas las configuraciones que aumentan el nivel de seguridad en dos módulos separados, uno para seguridad local y otro para seguridad de redes, además se codifica un tercer módulo para realizar las comprobaciones de integridad de datos.

Módulo de seguridad local

En este módulo se agrupan configuraciones que aporten ventajas en la seguridad local del sistema.

Ficheros Anexos
/etc/puppet/modules/local_security/manifests/init.pp

Clase local_security::password

En el fichero login.defs se especifican los parámetros de sistema a nivel de autenticación. en este caso sólo tendremos que asegurarnos que se está utilizando como método de cifrado SHA512, esto se especifica en la directiva ENCRYPT_METHOD SHA512

También se ha modificado la longitud mínima de password de cinco a ocho caracteres, se modifica el siguiente parámetro de configuración PASS_MIN_LEN 8.

En este caso para este fichero no existe lente de Augeas para poder modificar ficheros de configuración desde puppet, cuando no existe este componente utilizaremos una aproximación que consiste en descargar un fichero de configuración genérico. Se pueden definir ficheros de configuración específicos para cada host, tan sólo añadiendo el fqdn del host destino al nombre del archivo.

```
/etc/puppet/modules/local_security/manifests/password.pp
# password.pp - Local Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Class setting min password length
# TODO
#  - Augeas lense for this file
#  - Check password cypher<F2>

class local_security::password{

       file {
              'login.defs':
                     ensure =>     file,
                     path   =>     '/etc/login.defs',
                     source =>     [ "puppet:///modules/secure/login.defs.
$::fqdn" ,
                                   "puppet:///modules/secure/login.defs"
                                   ],
       }
```

}

Ficheros Anexos
/etc/puppet/modules/local_security/manifests/password.pp
/etc/puppet/modules/local_security/files/login.defs

Clase local_security::defaultrunlevel

Modificamos el valor del runlevel por defecto, en este caso a tres. No nos interesa que arranque en level cinco ya que tener el servidor X levantado únicamente iría en detrimento del rendimiento de la máquina. Para esta modificación sí se confía en Augeas para modificar el parámetro en el fichero /etc/inittab

Ficheros Anexos
etc/puppet/modules/local_security/manifests/defaultrunlevel

Clase local_security::sysinit

Se modifican tres parámetros del proceso init, que suponen diferentes riesgos de seguridad.

- PROMPT=no, con esto se consigue deshabilitar el inicio interactivo de servicios del sistema. Esta opción habilitada permitiría a cualquiera con acceso físico al servidor deshabilitar servicios durante el arranque de sistema sin necesidad de autenticarse.
- SINGLE=/sbin/sulogin. Si durante el arranque grub no dispone de password es posible pasar parámetros al kernel para arrancar en modo mantenimiento. Por defecto este modo de arranque no pide ningún tipo de autenticación para obtener una shell con privilegios de root. Con este parámetro conseguimos que al arrancar de este modo solicite el password de root para acceder al sistema.
- ACTIVE_CONSOLES=/dev/tty[1-2], se baja la cantidad de procesos getty de seis a dos para liberar recursos de sistema.

Todas estas configuraciones se realizan en puppet utilizando el módulo de Augeas para modificar configuraciones.

Ficheros Anexos
/etc/puppet/modules/local_security/manifests/sysinit.pp

Clase local_security::term

Por defecto no viene establecido ningún timeout sobre las consolas abiertas, es habitual abandonar un puesto de trabajo sin bloquear la sesión o incluso encontrarse consolas físicas con sesiones de root abiertas en un CPD. Para evitar que se tenga acceso a la cuenta privilegiada durante estas situaciones se ha de definir un tiempo máximo de inactividad en terminal tras el cual se cerrará la sesión automáticamente.

Para ello se crea el fichero /etc/profile.d/os-security.sh con los parámetros TMOUT a 900 segundos trás los cuales se desconectará automáticamente la sesión.

Ficheros Anexos
/etc/puppet/modules/local_security/manifests/term.pp

Clase local_security::extstorage

Para nada interesa que alguien con acceso físico al CPD tenga oportunidad de montar unidades de almacenamiento externas en los servidores, bien sea para extraer información sensible como para introducir software malicioso en la máquina sin dejar registros en firewalls o proxies.

Para ello desactivaremos la capacidad de montar dispositivos de almacenamiento externo impidiendo que se carguen los módulos del kernel que permiten utilizar dispositivos de USB y FireWire.

Nos aseguramos que en el directorio /etc/modprobe.d se encuentran dos ficheros que incluyan la directiva blacklist seguida del nombre del módulo a eliminar, en nuestro caso usb-storage y firewire_ohci

Ficheros Anexos
/etc/puppet/modules/local_security/manifests/extstorage.pp

Clase local_security::fsmount

Se ha seguido un esquema de particionamiento acorde al FHS, por lo que ahora procederemos a asegurar las particiones de /var, /tmp y /home.

En el fichero /etc/fstab se indican las siguientes opciones de montaje para estos tres volúmenes:

- Noexec, impide la ejecución de binarios en estas particiones. Evitamos así que un usuario suba un exploit a su home por ejemplo y lo ejecute. (También se han de eliminar las herramientas de compilación del sistema).
- Nosuid, evitamos que se establezcan los bits de suid y sgid en ficheros, estos bits establecen los permisos efectivos de un fichero a los de su propietario o grupo primario, en lugar de los del usuario que ejecuta el fichero y entrañan un riesgo de seguridad importante.
- Nodev, evitamos que se puedan utilizar estas rutas como puntos de montaje de otros sistemas de ficheros.

El fichero que establece los puntos de montaje es muy específico para cada sistema, la utilidad de gestionarlo a través de Puppet, más que simplificar la administración sería obtener un punto centralizado para realizar la misma. Además, se trata de reflejar mediante manifiestos todas las modificaciones realizadas en una

instalación base del sistema, de tal forma que simplifique la restauración de una determinada instalación.

A la hora de desarollar el manifiesto de puppet se utiliza Augeas para modificar los parámetros del fichero /etc/fstab

Ficheros Anexos
/etc/puppet/modules/local_security/manifiests/fsmount.pp

Clase local_security::issue

En la mayoría de sistemas al tratar de iniciar sesión viene por defecto un issue/issue.net dando la bienvenida al sistema e indicando tipo de sistema operativo y versión. Esto presenta dos riesgos importantes y habitualmente no tenidos en cuenta.

En primer lugar se ha de evitar dar toda la información posible del sistema a los posibles atacantes. Información del software instalado y versión del mismo, esto facilita realizar fingerprinting de la máquina y la búsqueda de exploits asociados a esa versión del sistema operativo / kernel. Es una buena práctica eliminar estos banners como medida de seguridad por ocultación.

El otro inconveniente que trataremos aquí es que la inmensa mayoría de estos banners por defecto incluyen la palabra “Welcome” lo que puede constituir un argumento de peso para la defensa a la hora de realizar una acusación penal por intrusión. Es por esto que se recomienda sustituir este banner por una advertencia legal que básicamente ha de indicar que sólo se permite el acceso y uso autorizado, advertir que se está realizando una monitorización activa de la máquina para detectar intentos de acceso no permitido y que esta información se utilizará con fines legales.

Utilizaremos por tanto el siguiente texto de advertencia:

```
-------------
W A R N I N G
-------------
THIS IS A PRIVATE COMPUTER SYSTEM.
This computer system including all related equipment, network devices
(specifically including Internet access), are provided only for
authorized use. All computer systems may be monitored for all lawful
purposes, including to ensure that their use is authorized, for
management of the system, to facilitate protection against unauthorized
access, and to verify security procedures, survivability and operational
security. Monitoring includes active attacks by authorized personnel and
their entities to test or verify the security of the system. During
monitoring, information may be examined, recorded, copied and used for
authorized purposes. All information including personal information,
placed on or sent over this system may be monitored. Uses of this
system, authorized or unauthorized, constitutes consent to monitoring of
this system. Unauthorized use may subject you to criminal prosecution.
Evidence of any such unauthorized use collected during monitoring may be
used for administrative, criminal or
other adverse action. Use of this system constitutes consent to
monitoring for these purposes.
```

Ficheros Anexos
/etc/puppet/modules/local_security/manifests/issue.pp
/etc/puppet/modules/local_security/files/issue

Clase local_security::cronrestrictions

Aplicaremos restricciones para que sólo el usuario root pueda programar tareas de cron, mediante la inclusión de todos los usuarios del sistema dentro del fichero /etc/cron.deny. El fichero /etc/cron.allow sólo contendrá una línea con el usuario root.

En este caso el guión de puppet especifica los comandos a ejecutar para copiar todos los usuarios presentes en /etc/password excepto root, al fichero de denegación de cron.

Ficheros Anexos
/etc/puppet/modules/local_security/manifests/cronrestrictions.pp

Clase local_security::ctrlaltdel

Cualquier acceso físico al servidor entraña un riesgo de denegación de servicio, por eso se ha de tener en cuenta el registro y control de acceso a CPD. En caso de disponer de una consola conectada al servidor, con sólo tener acceso a un teclado podría lograrse reinicio de la máquina al pulsarse la combinación de teclas ctrl+alt+del.

En nuestra configuración cambiaremos el comportamiento de este trap para que si se llega a pulsar esta combinación se guarde un registro en la bitácora del sistema que será monitorizado por Nagios. Como no se dispone de una lente de Augeas para este tipo de archivo, se distribuirá desde el servidor de ficheros una copia maestra personalizable para cada nodo con tan solo agregar su fqdn al nombre del fichero.

Para la monitorización se utilizará la comprobación check_log. A este plugin se le indicará un fichero de log a monitorizar y una ruta donde almacenar una copia de este fichero. Cada vez que se ejecute comprobará las diferencias entre estos dos archivos para ver si la expresión indicada ha aparecido en el archivo más reciente. En caso de encontrarla emitirá la alerta correspondiente.

Ficheros Anexos
/etc/puppet/modules/local_security/manifests/ctrlaltdel.pp
/etc/puppet/modules/local_security/files/ctrl-alt-delete.conf
/etc/puppet/modules/local_security/manifests/ctrlaltdel/nrpe.pp

Módulo se seguridad de red

Se desarrolla un módulo de seguridad de red que cubrirá todas aquellas mejoras sobre el sistema relacionadas con prevenir o mitigar ataques recibidos desde la red.

Ficheros Anexos
/etc/puppet/modules/network_security/manifests/init.pp

Clase network_security::sysctl

A través de la interfaz para parametrizar el comportamiento del kernel que supone sysctl.conf tocaremos diversas opciones que incrementen la seguridad de red.

Se modificarán las siguientes opciones:

Verificación de ruta de origen, mediante estás opciones se protege de spoofs de IP en la red, ya que no se aceptarán rutas de origen.

```
net.ipv4.conf.all.rp_filter
net.ipv4.conf.all.accept_source_route
```

No se aceptan redirecciones de red, se confía en la tabla de enrutamiento configurada para el direccionamiento y routing.

```
net.ipv4.conf.all.accept_redirects
```

Protección contra SYN flood, restringimos el número de respuestas ante una gran cantidad de paquetes SYN.

```
net.ipv4.tcp_syncookies 1
net.ipv4.tcp_synack_retries 2
```

Configuramos la red para que ignore cualquier ping de broadcast, lo que podría provocar saturación o ataques dirigidos contra un host al realizar IP snooping.

```
net.ipv4.icmp_echo_ignore_broadcasts 1
```

Ignoramos paquetes forjados en interfaces de red

```
net.ipv4.icmp_ignore_bogus_error_responses 1
```

Se registra cualquier paquete con flags inconsistentes.

```
net.ipv4.conf.all.log_martians 1,
```

Deshabilitamos sysrq, una funcionalidad de debug que permite mediante la combinación de teclas AltGr+BloqDisplay realizar diferentes tareas como sincronizar discos, enviar señal TERM a todos los procesos o reiniciar el servidor.

```
kernel.sysrq = 0
```

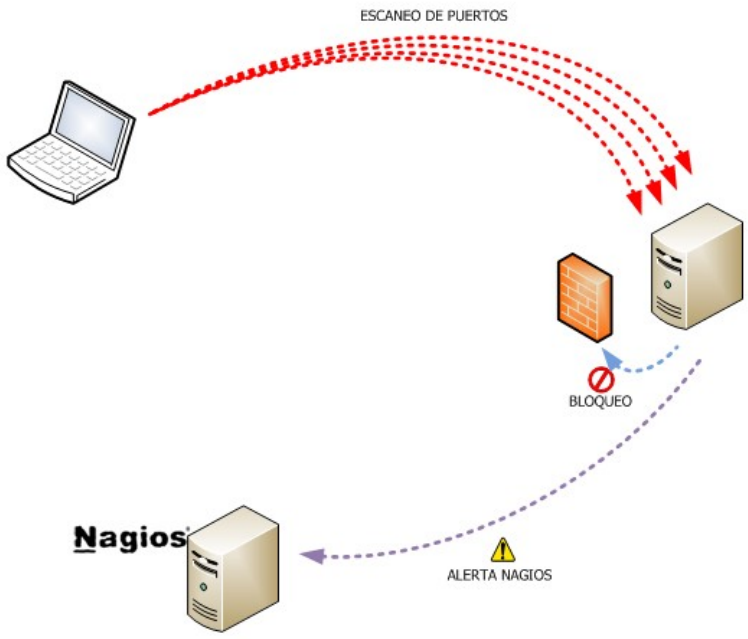
Se codifica un guión de puppet que monitoriza y mantiene mediante Augeas los parámetros del fichero /etc/sysctl.conf.

Ficheros Anexos
/etc/puppet/modules/network_security/manifests/sysctl.pp

Class network_security::portsentry

Portsentry es un software desarrollado por Psionic, se trata de un wrapper TCP destinado a detectar escaneo de puertos. Funciona levantando listeners en varios puertos específicos y ante la recepción de un paquete en ese puerto deniega la comunicación contra el origen.

Añade el host a /etc/hosts.deny y realiza acciones específicas en el firewall, además guarda un mensaje en la bitácora de sistema indicando que se ha detectado este comportamiento en la red.



Se ha descontinuado el desarrollo de este software al ser comprada la empresa que lo mantenía por Cisco, el paquete sigue disponible en sourceforge, descargamos el código fuente y lo compilamos.

Al tratar de compilarlo aparece un error en portsentry.c corregimos el mismo y se genera el siguiente diff:

```
diff portsentry.c portsentry.c.new
1584,1585c1584
<   printf ("Copyright 1997-2003 Craig H. Rowland <craigrowland at users
dot
< sourceforget dot net>\n");
---
>   printf ("Copyright 1997-2003 Craig H. Rowland <craigrowland at users
dot sourceforget dot net>\n");
```

Se modifica el fichero de configuración portsentry_conf.h para estandarizar las rutas de los ficheros de configuración de /usr/local a /etc

```
diff portsentry_config.h.orig portsentry_config.h
25c25
< #define CONFIG_FILE "/usr/local/psionic/portsentry/portsentry.conf"
---
> #define CONFIG_FILE "/etc/portsentry/portsentry.conf"
```

Una vez tenemos los binarios compilados modificamos el fichero de configuración para adecuarlo a NetFilter, tiene líneas de configuración para ipchains y ifuw pero no está actualizado para utilizar las herramientas de espacio de usuario iptables de la rama del Kernel 2.6, añadimos la siguiente línea al fichero de configuración.

```
KILL_ROUTE="iptables -A INPUT -s $TARGET$ -j DROP"
```

Añadir esta línea provoca que Netfilter haga drop de todas las conexiones establecidas por el origen, esto mantendrá todas las sesiones TCP abiertas hasta que expire el timeout.

Además modificamos los ficheros de históricos en este archivo de configuración para que apunten al directorio de configuración estandarizado /etc/portsentry:

```
diff portsentry.conf /etc/portsentry/portsentry.conf
83c83
< IGNORE_FILE="/usr/local/psionic/portsentry/portsentry.ignore"
---
> IGNORE_FILE="/var/spool/portsentry/portsentry.ignore""
85c85
< HISTORY_FILE="/usr/local/psionic/portsentry/portsentry.history"
---
> HISTORY_FILE="/var/spool/portsentry/portsentry.history"
87c87
< BLOCKED_FILE="/usr/local/psionic/portsentry/portsentry.blocked"
---
> BLOCKED_FILE="/var/spool/portsentry/portsentry.blocked"
217a218,223
> #
> # Added iptables sintax por netfilter
> # Set to drop if you want to timeout tpc sessions or Reject to send
RST
> #
```

```
> #KILL_ROUTE="iptables -A INPUT -s $TARGET$ -j REJECT"
> KILL_ROUTE="iptables -A INPUT -s $TARGET$ -j DROP"
```

Una vez ya tenemos el binario y el fichero de configuración, generamos a partir del script de inicio del demonio atd un script que arranque y pare el servicio. La ruta de destino para este fichero es /etc/rc.d/init.d/portsentry.

Con estos tres ficheros, el binario, el fichero de configuración y el script de arranque de init.d, generamos un fichero spec y creamos el rpm correspondiente. El proceso de generación de ficheros rpm y codificación del spec se encuentra fuera del ámbito de este documento.

Una vez generado el rpm ya lo podemos añadir a nuestro repositorio y ejecutar de nuevo createrepo para refrescar los índices del mismo.

A partir de este punto ya podemos generar el catálogo de puppet para la instalación y configuración del paquete.

También se configura la instalación correspondiente del comando NRPE para monitorizar mediante Nagios todos los escaneos de puertos detectados por Portsentry y registrados en /var/log/messages.

Ficheros Anexos
/root/porsentry.spec
/etc/portsentry/portsentry.conf
/etc/rc.d/init.d/portsentry
/etc/puppet/modules/network_security/manifests/portsentry.pp
/etc/puppet/modules/network_security/manifests/portsentry/nrpe.pp

Clase network_security::ssh

El demonio SSH viene configurado y levantado por defecto en todas las instalaciones de CentOS, se realizarán una serie de modificaciones en su fichero de configuración.

Se cambia el puerto en el que escucha normalmente a uno menos habitual (10022), de esta forma evitamos tener que revisar incontables líneas de log con intentos de acceso por gusanos tratando de hacer login con usuarios genéricos. Se deshabilita SSHv1 y se restringe únicamente a la versión dos, la única considerada segura. Se desactivan también la redirección de conexiones gráficas, la autenticación basada en host y se configuran los mensajes legales que se mostrarán durante el inicio de sesión. Como opción podríamos distribuir las claves públicas que nos interesen a través de puppet y basar la autenticación en clave pública en lugar de en password, distribuyendo el fichero authorized_keys.

El fichero de configuración que se servirá es el original de la instalación del openssh-server, con las siguientes modificaciones

```
diff /etc/ssh/sshd_config modules/ssh/files/sshd_config
13c13
< #Port 22
---
> Port 10022
56c56
< #HostbasedAuthentication no
---
> HostbasedAuthentication no
61c61
< #IgnoreRhosts yes
---
> IgnoreRhosts yes
108c108
< X11Forwarding yes
---
> X11Forwarding no
115c115
< #UsePrivilegeSeparation yes
---
> UsePrivilegeSeparation yes
128c128
< #Banner none
---
> Banner /etc/issue.net
```

También se genera el fichero de configuración para NRPE que define el comando que realizará la comprobación sobre el fichero /var/log/secure buscando intentos de acceso fallidos y notificando la correspondiente alerta Nagios.

Ficheros Anexos
/etc/puppet/modules/network_security/ssh.pp
/etc/puppet/modules/network_security/ssh/nrpe.pp

Clase network_security::iptables

Se levanta en cada instalación del servidor en sistema de filtrado de paquetes Netfilter, activamos el firewall local para tener el control sobre el tráfico entrante y saliente de los servidores.

La política a aplicar será denegar todas aquellas conexiones que no se hayan autorizado específicamente con una regla, esto provoca una mayor carga de trabajo en la administración del firewall pero es la opción más seguro y deseable.

Puppet dispone de una limitación a la hora de manejar iptables, no es factible notificar cambios en el estado del firewall sin que se reinicie automáticamente con cada ejecución del demonio. La integración de Puppet con iptables es un tema candente en los PuppetLabs y existen varias lineas de desarrollo abiertas.

Como aproximación definiremos el firewall en el servidor maestro y lo enviaremos a los clientes en la ruta /etc/sysconfig/iptables, crearemos también una dependencia de este fichero hacia el servicio iptables, para refrescar el firewall con los cambios.

Ficheros Anexos
/etc/puppet/modules/network_security/manifests/iptables.pp
/etc/puppet/modules/network_security/files/iptables

Módulo de comprobación de integridad

Con este módulo se pretenden añadir comprobaciones de integridad de binarios y librerías en los sistemas que controlemos, de esta manera tratamos de protegerlos frente a intrusiones consumadas que pretendan perpetuarse con la instalación de un rootkit.

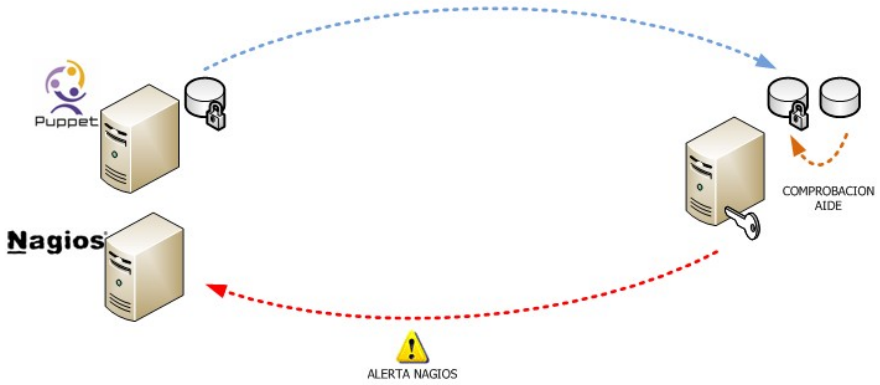
Ficheros Anexos
/etc/puppet/modules/integrity_check/manifests/init.pp

Clase integrity_check::aide

AIDE (Advanced Intrusion Detection Environment) es una herramienta de comprobación de integridad de ficheros. Crea una base de datos de los ficheros del sistema y de información de sus atributos extendidos, tras esto es capaz de comparar con diversos algoritmos la integridad contrastándola con la base de datos.

Debido a que regenerar esta base de datos tras comprometer el sistema sólo necesitaría de un simple comando y pasaría completamente desadvertido, la genialidad que nos ofrece puppet es que podemos guardar esta base de datos en el servidor maestro y asegurarnos con esta copia externa la integridad del mismo. Para ello se ha de ejecutar el script puppet_ic_sync y se creará una nueva base de datos de AIDE y se subirá por ssh al servidor.

Evidentemente no es confiable una autenticación por clave pública en sentido cliente servidor, por lo que nos deberemos autenticar con las credenciales del master al realizar la sincronización.



Se genera el script de Puppet que se encargará de instalar AIDE además de generar y mantener la base de datos sincronizada con el servidor, con el manifiesto de puppet nos aseguramos que el paquete está instalado. También se codifica el script que se encarga de subir la base de datos de firmas de AIDE al servidor de puppet mediante SSH.

Dado que la carga de generar la base de datos de firmas es muy elevada, se programa una tarea cron en el guión de puppet para que se realice la comprobación de integridad una vez por semana.

Se genera también el correspondiente guión para la instalación del comando NRPE que se encargará de comprobar el fichero /var/log/aide/aide.log y notificar a la sonda nagios en caso de que haya una inconsistencia con el catálogo bajado del servidor.

Se utiliza el check_log que contrasta el log con un histórico para que no se repitan las alarmas y busca expresiones regulares dentro del mismo.

Ficheros Anexos
/etc/puppet/modules/integrity_check/manifests/aide.pp
/etc/puppet/modules/integrity_check/manifests/aide/nrpe.pp
/etc/puppet/modules/integrity_check/files/aide_db_sync

Clase integrity_check::rkhunter

RKHunter es una herramienta de seguridad que se encarga de buscar rootkits, backdoors y exploits comparando la suma de integridad de diferentes binarios esenciales, buscando firmas de exploits conocidos y comparándolos contra una base de datos local. Es importante ejecutar periódicamente estas comprobaciones, así como mantener la base de datos de firmas actualizada. Este paquete se encuentra disponible desde los repositorios extendidos por lo que no deberemos realizar ninguna operación más que instalarlo desde el gestor de paquetes de la distribución.

En el guión de Puppet se especifica que el paquete se encuentre instalado, así como su dependencia de cron, se genera una tarea programada semanal que ejecutará el siguiente comando:

```
/usr/bin/rkhunter --update && /usr/bin/rkhunter -c --cronjob 2>&1
```

Así aseguramos antes de cada ejecución que se ha actualizado la base de datos de firmas y estamos protegidos contra todos los exploits detectados por RKhunter. También se genera el correspondiente fichero de configuración NRPE para monitorizar los resultados de la auditoría.

Ficheros Anexos
/etc/puppet/modules/integrity_check/manifests/rkhunter.pp
/etc/puppet/modules/integrity_check/manifests/rkhunter/nrpe.pp

Clase integrity_check::lynis

Lynis es una herramienta de seguridad Unix, realiza tareas de auditoría automatizada de seguridad, administración de parches de software y scanner de vulnerabilidades y malware. Además de como herramienta de seguridad también

recopila información general del sistema, paquetes instalados y errores de configuración.

Se descarga el paquete de software Lynis desde su web y el fichero spec para la creación del paquete rpm que distribuiremos en nuestro repositorio local.

```
http://www.rootkit.nl/files/lynis-1.2.9.tar.gz
http://www.rootkit.nl/files/lynis.spec
```

Corregimos en el fichero spec el número de versión y eliminamos una referencia perdida a un fichero que ya no existe (afortunadamente ya que el nombre de fichero es “TODO”).

Generamos el siguiente diff frente al spec:

```
23c23
< Version:          1.1.5
---
> Version:          1.2.9
91c91
< %doc CHANGELOG FAQ LICENSE README TODO
---qq
> %doc CHANGELOG FAQ LICENSE README
```

Movemos el rpm generado a la ruta de nuestro repositorio /var/www/html/repo y actualizamos los índices del mismo con createrepo.

En el guión de puppet se define el paquete a instalar y el trabajo de cron que se ejecutará semanalmente. También se define la clase de monitorización integrity_chec::lynis::nrpe, con la comprobación de Nagios. Se utiliza de nuevo check_log y se verifican los ficheros de log buscando expresiones regulares.

Ficheros Anexos
/etc/puppet/modules/integrity_check/manifests/lynis.pp
/etc/puppet/modules/integrity_check/manifests/lynis/nrpe.pp

Plantilla de servidor AntiSpam

Se diseña un sistema frontend de correo para ubicarlo en DMZ que actúe como mail exchanger de la infraestructura, recepcionará el correo directamente de internet y tras realizar un filtrado antispam y antivirus transmitirá la correspondencia a un MDA situado en una DMZ de menor nivel, quién se encargará de catalogar y entregar los mails en los buzones de los usuarios.

Se confía en Postfix para esta tarea. Este servidor de correo nace como alternativa a Sendmail, se diseñó teniendo en mente las fallas de seguridad sufridas por Sendmail y desde su diseño se tiene muy en cuenta la seguridad del sistema de correo.

Se compone de varios demonios que se encargan de la distribución puntual de correos entre las diferentes colas que lo componen, son gestionados por un demonio master que los va lanzando bajo demanda. Además ofrece la posibilidad de enjaular estos demonios de tal forma que el compromiso de uno de ellos no afecte al resto ni al sistema.

Como sistema de seguridad de correo utilizaremos MailScanner, este software se encargará de aplicar los filtros definidos sobre cada mail que se reciba. El funcionamiento es sencillo, hemos de configurar Postfix para que todo mail entrante lo pase a la cola hold. MailScanner se encargará de recoger estos mails y filtrarlos. Tras esto o bien los deja en la cola Active para que continúen su flujo normal dentro del MTA o los marca como mail no deseado con lo que podrían ponerse en cuarentena o ser borrados directamente.

MailScanner dispone de numerosos y avanzados sistemas de filtrado, evitando de este modo intentos de phising, archivos adjuntos peligrosos, etc.

Para el filtrado de spam se utilizará el aclamado filtro de correo SpamAssassin, aplica filtros de spam que se mantienen actualizados por internet a los correos entrantes. Además dispone de un filtro bayesiano al que podremos ir alimentando con los correos marcados por los usuarios como spam y que no han sido detectados por el filtro, para que el propio sistema aprenda de sus errores y mejore su nivel de eficiencia.

SpamAssassin puntúa cada mail recibido en función de cabeceras, contenido, etc. Este puntaje es incluido en la cabecera del mail y posteriormente será evaluado por MailScanner para decidir si el correo es legítimo o spam.

También se realiza un filtrado de virus y malware a cargo de ClamAV, un conocido software de código libre diseñado para ser utilizado principalmente como antivirus en sistemas de correo electrónico.

ClamAV se puede ejecutar bajo demanda por MailScanner o como demonio propio comunicándose con éste por medio de un socket. Por motivos de rendimiento se

ha decidido lanzar una instancia propia de ClamAV estableciéndose un socket tipo Unix para el intercambio de información.

Por tanto para configurar este sistema hemos de configurar cuatro instalaciones de productos distintos y después interconectarlos entre si.

Ficheros Anexos
/etc/puppet/modules/aspam/manifests/init.pp

Postfix

Clase aspam::postfix

Realizaremos una instalación de Postfix básica en la que configuraremos los parámetros necesarios para que el servidor se ubique dentro de internet mediante su FQDN, conozca cuales son sus redes permitidas, su nombre de host y de dominio y a qué nodo interno ha de enviar todo el mail legítimo recibido de internet.

Mediante Augeas se configuran los siguientes parámetros que se pasarán a la clase desarrollada:

- myhostname
- mynetworks
- relaydomains
- relayhost
- mydomain

También se modifica el banner de saludo del servidor para que indique el dominio al que pertenece y del que es mail exchanger, esto aporta la ventaja de que no estamos saludando a todos nuestros clientes diciéndoles que software y versión estamos utilizando.

Se imponen restricciones en el saludo y al remitente del mensaje, con esto conseguimos que el primer demonio de Postfix que recibe los mensajes rechaze aquellos malformados y de servidores sin un nombre de dominio adecuado.

Además se deshabilita el comando vrfy en el servicio smtpd, este comando puede ser utilizado por fuerza bruta para obtener información de los posibles destinatarios de correo del servidor.

También implementamos una lista negra de relay, utilizaremos la lista Zen de Spamhaus.org, ésta incorpora cuatro sublistas negras de este mismo dominio. Mediante consultas DNS sobre el nombre del servidor que nos está intentando transferir correo podemos saber si está incluido en listas públicas de fuentes conocidas de spam:

- SBL, Spamhaus Block List. Listas de IP's que han sido detectadas enviando UBE.
- XBL, Exploits Block List, esta lista incluye direcciones que se han identificado como emisoras de spam por parte de troyanos, gusanos, etc. Incorpora datos de otras dos importantes listas negras: Abuseat.org y njabl.org.
- PBL, Policy Block List, incluye rangos de direcciones IP en las que no se espera que esté corriendo un servicio smtp. Necesita de la colaboración de ISP's por lo que no es de las más confiables.
- DBL, Domain Block List, esta lista incluye dominios que han sido detectados en mensajes de spam, idealmente está diseñada para la etapa de análisis de contenido.

Ficheros Anexos
/etc/puppet/modules/aspam/manifests/postfix.pp

MailScanner

Clase aspam::ms

Para la instalación de MailScanner no existe ningún paquete de distribución, desde la página oficial únicamente podemos descargar un fichero comprimido con las fuentes que incluye los src.rpms de las dependencias. Contiene un script que se encarga de resolver estas dependencias, las compila y genera entonces un fichero rpm con el software y lo instala.

El trabajo realizado aquí incluye identificar esas dependencias que resuelve el script de instalación incluido con el paquete y satisfacerlas mediante Puppet. Así generamos un paquete para MailScanner instalable de forma reproducible en varias máquinas sin necesidad de ejecutar el script de instalación. Las dependencias encontradas son módulos de Perl disponibles en los repositorios de la distribución[13][14]

Uno de los puntos a tener en cuenta a la hora de trabajar con Puppet es que todas las plantillas que definamos se ejecutarán periódicamente por el demonio corriendo en el cliente.

Podríamos plantear el descargar un paquete de software desde el servidor de ficheros integrado y delegar en el demonio la ejecución del script de instalación. Estaríamos cometiendo un fallo imperdonable ya que el resultado obtenido sería un demonio que cada treinta minutos reinstala el software.

MailScanner dispone de un directorio conf.d donde podremos incluir todas las personalizaciones de configuración que realicemos sobre el fichero base, si bien por una limitación del producto no podemos definir en este directorio algunas directivas que se utilizan como fuente en otras opciones de configuración. Estas directivas son

13 Mantenimiento evolutivo: Instalación frontend MailWatcher
14 Mantenimiento evolutivo: Montar ramdiskfs para en el directorio de trabajo de MailScanner para mejorar el rendimiento al analizar adjuntos comprimidos.

parámetros como el hostname y todas ellas se corresponden con la regex %(.*)%. Debido a esto los ficheros de configuración se establecerán en el servidor maestro y se descargará un fichero distinto para cada servidor cliente.

Las opciones de configuración que se personalizan en el fichero /etc/MailScanner/MailScanner.conf serán:

- Configuración de los informes en español.
  ```
  %report-dir% = /etc/MailScanner/reports/es/
  ```
- Datos sobre la empresa a la que pertenece el sistema de correo.
  ```
  %org-name% = [Nombre Empresa]
  %org-long-name% = Your Organisation Name Here
  %web-site% = www.your-organisation.com
  ```

La configuración que conecta MailScanner con el resto de módulos del sistema las ubicaremos en un fichero de configuración en /etc/MailScanner/conf.d/MailScanner.prefs.conf, este fichero será el mismo para todos los sistemas y definimos el siguiente contenido en el que se indica el usuario con el que lanzará el servicio, las colas de entrada y salida de mail, antivirus y rutas, permisos, etc.

```
Run As User = postfix
Run As Group = postfix
Incoming Queue Dir = /var/spool/postfix/hold
Outgoing Queue Dir = /var/spool/postfix/incoming
MTA = postfix
Incoming Work User =
Incoming Work Group = clam
Incoming Work Permissions = 0660
Quarantine User = root
Quarantine Group = apache
Quarantine Permissions = 0660
Virus Scanners = clamd
Clamd Socket = /var/run/clamav/clamd.sock
Quarantine Whole Message = yes
Log Spam = yes
SpamAssassin User State Dir =/var/spool/MailScanner/spamassassin
```

Ficheros Anexos
/etc/puppet/modules/aspam/manifests/ms.pp
/etc/puppet/modules/aspam/files/MailScanner.conf
/etc/puppet/modules/aspam/files/MailScanner.prefs.conf

ClamAV

Clase aspam::clamd

La instalación del antimalware ClamAV es realmente sencilla, además de porque el paquete está disponible en los repositorios extendidos de la distribución, porque sólo tendremos que tocar un parámetro de configuración. El guión de Puppet se encargará entonces de asegurarse de que tanto paquete como dependencias (cron y comando file) estén satisfechas.

En las opciones de configuración por defecto ya se establece el socket que hemos configurado previamente en MailScanner, únicamente se ha de añadir el usuario bajo el que corre el demonio clamd y añadirlo al grupo postfix para que tenga acceso al directorio incoming de MailScanner /var/spool/MailScanner/incoming/.

También se configura la actualización de la base de datos de firmas de malware mediante el comando que proporciona el paquete, freshclam. Se programa una tarea de cron para que se ejecute la actualización diariamente a la una de la madrugada.

Clamd es el único software que ha generado alertas contra el vector de seguridad de SELinux, tratando de acceder a los directorios de colas de MailScanner, se ha generado un nuevo módulo SELinux que permite el acceso por parte de este proceso a esta cola. Para evitar complicar la instalación no se genera un contexto de seguridad común para ambos procesos, sólo se otorgan privilegios de acceso y lectura sobre estos ficheros.

Para generar el módulo primero cambiaremos el comportamiento de SELinux de enforcing a permissive de tal forma que se loguen todas las colisiones contra el vector de acceso por el demonio audit. Tras cambiar a permissive encolamos un mail y vemos las colisiones en el fichero /var/log/audit/audit.log, nos interesarán todos los mensajes marcados por la cadena AVC.

```
type=AVC msg=audit(1324694054.476:107): avc: denied { read } for
pid=12204
comm="clamd" name="12154" dev=dm-4 ino=7830
scontext=unconfined_u:system_r:clamd_t:s0
tcontext=unconfined_u:object_r:var_spool_t:s0
tclass=dir
type=AVC msg=audit(1324703243.566:233): avc: denied { getattr } for
pid=14582
comm="clamd" path="/var/spool/MailScanner/incoming/14516/1/neicar.com"
dev=dm-4 ino=2464 scontext=unconfined_u:system_r:clamd_t:s0
tcontext=unconfined_u:object_r:var_spool_t:s0 tclass=file
type=AVC msg=audit(1324703626.383:239): avc: denied { read } for
pid=14931
comm="clamd" name="neicar.com" dev=dm-4 ino=2466
scontext=unconfined_u:system_r:clamd_t:s0
tcontext=unconfined_u:object_r:var_spool_t:s0
tclass=file
type=AVC msg=audit(1324781266.918:393): avc: denied { open } for
pid=22670
comm="clamd" name="neicar.com" dev=dm-4 ino=2814
scontext=unconfined_u:system_r:clamd_t:s0
tcontext=unconfined_u:object_r:var_spool_t:s0
tclass=file
```

Copiamos todos estos mensajes y utilizaremos la herramienta audit2allow para generar las políticas adecuadas, ejecutamos para ello el siguiente comando sobre el fichero de colisiones.

```
audit2allow -m clamd < clamd.avc > clamd.te
```

En el contenido del tipo de seguridad generado clamd.te podemos ver los permisos específicos de SELinux para los cuales compilaremos el módulo.

```
module clamd 1.0;
require {
type var_spool_t;
type clamd_t;
class dir read;
class file { read getattr open }; }
#============= clamd_t ==============
allow clamd_t var_spool_t:dir read;
allow clamd_t var_spool_t:file open;
allow clamd_t var_spool_t:file { read getattr };
```

Vemos que tan sólo es necesario permitir lectura, apertura y obtener los atributos de los ficheros de tipo var_spool_t al tipo clamd_t. Compilamos ahora el módulo y lo cargamos en SELinux.

```
#make -f /usr/share/selinux/devel/Makefile clamd.pp
#semodule -i clamd.pp
```

Ficheros Anexos
/etc/puppet/modules/aspam/manifests/clamd.pp
/root/clamd.avc
/root/clamd.te

SpamAssassin

Class aspam::spamassassin

La instalación de este software es realmente sencilla, sólo nos aseguraremos de que el paquete se encuentra instalado desde repositorio y de generar una tarea de cron que actualice los filtros de spam ejecutando el comando sa-update diariamente.

Ficheros Anexos
/etc/puppet/modules/aspam/manifests/spanassassin.pp

Prueba de funcionamiento antispam y antivirus

Para testear el correcto funcionamiento del sistema antispam haremos uso de las herramientas de benchmark que vienen con Postfix: smtp-source y smtp-sink. Generamos con smtp-source quince threads que envíen un total de mil mails a nuestro sistema de correo. Podemos ver en el fichero de log /var/log/maillog que se están procesando correctamente los correos y se entregan en el buzón local de root.

```
smtp-source -s 15 -m 1000 -c -f root@master.test -t root@nodos.test
192.168.0.230:25
```

Para probar el sistema de filtrado antispam utilizaremos GTUBE, se trata de una cadena de texto predefinida que ha de ser detectada por todos los sistemas de filtrado

antispam, su finalidad es probar el correcto funcionamiento del mismo. La cadena es como sigue:

```
XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X
```

Enviamos un mail con esta cadena tecleando el siguiente comando:

```
# mail root@centos.test
Subject: test gtube
XJS*C4JDBQADN1.NSBN3*2IDNEN*GTUBE-STANDARD-ANTI-UBE-TEST-EMAIL*C.34X
EOT
```

Y observamos en los logs de correo como es detectado correctamente.

```
Dec  5 13:26:53 centos postfix/pickup[3517]: 747B91360: uid=0
from=<root>
Dec  5 13:26:53 centos postfix/cleanup[3558]: 747B91360: hold: header
Received: by centos.test (Postfix, from userid 0)??id 747B91360; Mon,  5
Dec 2011 13:26:53 +0100 (CET) from local; from=<root@centos.test>
to=<root@centos.test>
Dec  5 13:26:53 centos postfix/cleanup[3558]: 747B91360: message-
id=<20111205122653.747B91360@centos.test>
Dec  5 13:26:54 centos MailScanner[3546]: New Batch: Scanning 1
messages, 692 bytes
Dec  5 13:26:54 centos MailScanner[3546]: Virus and Content Scanning:
Starting
Dec  5 13:27:02 centos MailScanner[3546]: Spam Checks: Starting
Dec  5 13:27:02 centos MailScanner[3546]: Message 747B91360.A1059 from
127.0.0.1 (root@centos.test) to centos.test is spam, SpamAssassin
(puntaje=1002.424, requerido 6, DKIM_ADSP_NXDOMAIN 0.80, GTUBE 1000.00,
NO_RELAYS -0.00, SUBJ_ALL_CAPS 1.62)
Dec  5 13:27:02 centos MailScanner[3546]: Spam Checks: Found 1 spam
messages
Dec  5 13:27:02 centos MailScanner[3546]: Spam Actions: message
747B91360.A1059 actions are store
Dec  5 13:27:03 centos MailScanner[3546]: Deleted 1 messages from
processing-database
```

Para la prueba del filtrado antivirus existe el mismo procedimiento con la cadena EICAR, una cadena de texto con la misma finalidad y que debe ser detectada por todos los antivirus.

```
X5O!P%@AP[4\PZX54(P^)7CC)7}$EICAR-STANDARD-ANTIVIRUS-TEST-FILE!$H+H*
```

Tras enviarlo de la misma manera que con GTUBE, podemos ver en los logs del sistema la detección del mismo por parte de ClamAV.

```
Dec  8 05:47:05 nodos MailScanner[9687]: Found 1 messages in the
Processing Attempts Database
Dec  8 05:47:05 nodos MailScanner[9687]: Using locktype = flock
Dec  8 05:49:29 nodos MailScanner[9687]: Making attempt 5 at processing
message E9D5E14D8.A7416
Dec  8 05:49:29 nodos MailScanner[9687]: New Batch: Scanning 1 messages,
679 bytes
Dec  8 05:49:30 nodos MailScanner[9687]: Virus and Content Scanning:
Starting
```

```
Dec  8 05:49:30 nodos MailScanner[9687]: Clamd::INFECTED::Eicar-Test-
Signature :: ./E9D5E14D8.A7416/
Dec  8 05:49:30 nodos MailScanner[9687]: Virus Scanning: Clamd found 1
infections
Dec  8 05:49:30 nodos MailScanner[9687]: Infected message
E9D5E14D8.A7416 came from 127.0.0.1
Dec  8 05:49:30 nodos MailScanner[9687]: Virus Scanning: Found 1 viruses
```

Plantilla de servidor VPN Roadwarrior

La funcionalidad requerida del sistema, es la de proporcionar un servicio de VPNs para la conexión de usuarios remotos a través de Internet. Dadas sus prestaciones y seguridad, el servicio se ha implementado mediante OpenVPN, utilizando una autenticación basada en certificados X.509.

Dado que el tema es complejo damos a continuación algunos detalles generales:

- El servicio de VPNs se instalará y configurará en el servidor para proporcionar conexiones encapsuladas mediante un transporte TCP: 1994
- Utilizamos por defecto dicho transporte porque evita la mayor parte de problemas de conexión si el usuario remoto está tras un firewall/proxy.
- Una vez conectado el usuario remoto recibirá una IP en una red 'filtrada' mediante un adaptador TUN, de forma que todo su tráfico será 'forzado' a pasar por la VPN, incluso el de acceso directo a Internet, por lo que el cliente quedará 'protegido' por la infraestructura de seguridad de la empresa (firewall, antivirus, ...)
- La autenticación de conexión se realiza mediante sendos certificados X.509 (uno en el servidor y otro en el cliente), y es mutua. El certificado del cliente esta a su vez protegido por un password por lo que incluso en caso de pérdida del fichero/portátil, mantenemos un nivel aceptable de seguridad.
- Una vez establecida la conexión todo el tráfico queda protegido en un canal TLS cifrado.

A partir de estas afirmaciones, es evidente que vamos a necesitar los siguientes elementos:

- Un certificado X.509 en el servidor y otro en cada cliente
- Necesitamos por tanto configurar una CA en el servidor.
- Una instalación de OpenVPN activa y configurada en el servidor.

Primero generamos el script Puppet que se encargue de instalar el paquete de OpenVPN y se asegure de que está corriendo y que está configurado para arrancarse con cada reinicio de la máquina.

Una vez hecha esta parte nos encargaremos de la generación de certificados para el cliente. Haremos uso de los scripts easy-rsa que se instalan con el paquete OpenVPN. Para cada servidor se generará un certificado de autoridad certificadora, que será la que firme el certificado del servidor y clientes.

Creamos una copia de los scripts en el repositorio de puppet, debido a que tenemos que sourcear las variables correspondientes al certificado para cada generación de nuevas claves para cliente, replicaremos los scripts y las configuraciones.

Copiamos el contenido de /usr/share/openvpn/easy-rsa/2.0/ a /etc/puppet/modules/openvpn/files/generic

Para cada cliente copiaremos este directorio generic a un directorio cuyo nombre coincida con el fqdn del servidor destino de los certificados generados. Copiamos la plantilla de configuración para openssl que viene en el directorio y la renombramos de openssl-1.0.0.cnf a openssl.cnf

En primer lugar editamos el fichero vars y lo cumplimentamos con los datos requeridos para la generación del certificado:

```
/etc/puppet/modules/openvpn/files/generic/vars
export KEY_COUNTRY="ES"
export KEY_PROVINCE="BCN"
export KEY_CITY="Barcelona"
export KEY_ORG="Organization Name"
export KEY_EMAIL="admin@example.org"
export KEY_EMAIL=mail@example.org
export KEY_CN=fqdn.server
export KEY_NAME=openvpnserver
export KEY_OU=IT
export PKCS11_MODULE_PATH=changeme
export PKCS11_PIN=1234
```

Tras esto importamos las variables de entorno a la shell en curso con el comando

```
#source vars
```

Generamos las claves y certificado de la autoridad certificadora, utilizaremos estos certificados para firmar los de servidor y cliente. Vemos que en el directorio keys se generan los ficheros ca.crt y ca.key. Podemos volcar el contenido del certificado con el comando:

```
#openssl x509 -in keys/ca.crt -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number:
            e6:53:5d:17:41:fe:2b:9c
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=ES, ST=BCN, L=Barcelona, O=Organization Name, OU=IT,
CN=fqdn.server/name=openvpnserver/emailAddress=mail@example.org
        Validity
            Not Before: Jan  9 12:27:24 2012 GMT
            Not After : Jan  6 12:27:24 2022 GMT
        Subject: C=ES, ST=BCN, L=Barcelona, O=Organization Name, OU=IT,
CN=fqdn.server/name=openvpnserver/emailAddress=mail@example.org
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
```

```
                Modulus:
                    00:b8:c7:76:17:9c:2d:9a:cc:1e:87:9c:4f:fc:d9:
                    80:98:85:29:3f:23:81:e9:35:0e:fc:9d:ce:e7:de:
                    bf:42:ec:56:6c:67:d6:f8:f7:a6:84:20:77:3f:eb:
                    f6:a7:18:b2:e2:d9:6a:04:99:b7:ce:65:69:b5:a6:
                    70:8e:98:49:c4:61:ae:cc:6b:39:67:e5:b9:50:33:
                    7e:31:59:57:f1:3f:b3:8f:47:56:1e:8a:05:1c:63:
                    4b:38:20:1b:44:1d:25:f7:8c:87:da:18:83:2f:d9:
                    c2:9c:fc:c6:9f:eb:fe:b4:6c:b6:7e:a3:13:e0:54:
                    6f:40:7f:84:d8:08:cd:2a:09
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Subject Key Identifier:
                15:0E:F2:60:36:F2:4D:85:4F:FB:4A:C5:1B:41:49:9D:BA:2F:F6
:4D
            X509v3 Authority Key Identifier:
                keyid:15:0E:F2:60:36:F2:4D:85:4F:FB:4A:C5:1B:41:49:9D:BA
:2F:F6:4D
                DirName:/C=ES/ST=BCN/L=Barcelona/O=Organization
Name/OU=IT/CN=fqdn.server/name=openvpnserver/emailAddress=mail@example.o
rg
                serial:E6:53:5D:17:41:FE:2B:9C

            X509v3 Basic Constraints:
                CA:TRUE
    Signature Algorithm: sha1WithRSAEncryption
        48:ba:75:c6:32:3a:06:45:f6:be:f3:dd:62:f1:67:92:0b:d6:
        de:8e:52:34:d0:87:1c:ef:36:40:62:2f:bd:a6:d9:18:3d:62:
        cb:92:5e:49:8a:41:a6:37:02:00:0c:59:39:95:88:09:a4:da:
        21:77:51:0d:22:47:00:c6:b4:b1:b6:92:20:0f:b1:4b:c5:a7:
        e4:a4:b7:1f:8c:e2:31:19:e4:67:c7:37:1b:93:4f:6c:c9:37:
        fe:1a:d8:b3:d4:da:61:0d:c6:46:f0:1b:9c:4e:c2:44:ac:43:
        95:01:70:e5:25:80:9c:a1:3a:54:a6:c0:58:a9:d2:fb:45:ca:
        3d:c9
```

Tras esto generamos el par de claves Diffie-Hellman utilizado para el intercambio seguro de certificados a la hora de establecer la conexión VPN.

```
#./build-dh
```

Vemos que se ha generado también en el directorio keys el fichero dh1024.pem

Por último se ha de generar el certificado para el servidor, para ello ejecutamos el script:

```
#./build-key-server server
```

Esto genera la clave server.key y la solicitud de firma de certificado correspondiente server.csr, el último paso del script consiste en la firma por parte de la CA de este csr. Podemos ver los campos del certificado con openssl:

```
#openssl x509 -in keys/server.crt -text
Certificate:
    Data:
        Version: 3 (0x2)
        Serial Number: 1 (0x1)
        Signature Algorithm: sha1WithRSAEncryption
        Issuer: C=ES, ST=BCN, L=Barcelona, O=Organization Name, OU=IT,
```

```
CN=fqdn.server/name=openvpnserver/emailAddress=mail@example.org
        Validity
            Not Before: Jan  9 12:31:59 2012 GMT
            Not After : Jan  6 12:31:59 2022 GMT
        Subject: C=ES, ST=BCN, L=Barcelona, O=Organization Name, OU=IT,
CN=server/name=openvpnserver/emailAddress=mail@example.org
        Subject Public Key Info:
            Public Key Algorithm: rsaEncryption
                Public-Key: (1024 bit)
                Modulus:
                    00:c6:58:ce:fd:71:19:c2:ff:03:99:3f:26:f7:bd:
                    b3:63:2a:7a:1e:f4:38:be:62:77:56:e1:f6:ac:58:
                    4d:4f:72:f0:9e:de:95:b9:ce:53:70:95:4c:0c:0e:
                    21:6c:31:c3:19:9e:91:8e:fe:9e:6b:6d:46:9d:be:
                    62:e8:38:8c:0c:52:c2:80:1d:65:02:48:e6:1c:31:
                    d6:6a:4a:4f:e4:c8:10:ed:7b:d7:cd:14:ee:20:e4:
                    57:38:2c:ef:10:46:d9:cb:c2:94:9b:ac:ac:41:07:
                    28:a2:57:cd:92:be:8a:b7:cc:ef:a2:90:7e:41:79:
                    d3:e2:47:5d:52:ed:d8:99:bf
                Exponent: 65537 (0x10001)
        X509v3 extensions:
            X509v3 Basic Constraints:
                CA:FALSE
            Netscape Cert Type:
                SSL Server
            Netscape Comment:
                Easy-RSA Generated Server Certificate
            X509v3 Subject Key Identifier:
                2E:58:23:08:30:6F:6C:4D:C5:24:7E:07:59:AE:95:7C:6E:38:AB
:20
            X509v3 Authority Key Identifier:
                keyid:15:0E:F2:60:36:F2:4D:85:4F:FB:4A:C5:1B:41:49:9D:BA
:2F:F6:4D
                DirName:/C=ES/ST=BCN/L=Barcelona/O=Organization
Name/OU=IT/CN=fqdn.server/name=openvpnserver/emailAddress=mail@example.o
rg
                serial:E6:53:5D:17:41:FE:2B:9C

            X509v3 Extended Key Usage:
                TLS Web Server Authentication
            X509v3 Key Usage:
                Digital Signature, Key Encipherment
    Signature Algorithm: sha1WithRSAEncryption
        b3:4a:09:2c:dc:67:7f:10:89:b4:2b:3a:6a:62:a1:b8:7e:cf:
        0b:a1:b9:69:a1:99:6c:5a:3d:86:87:78:db:30:7e:b6:d5:b9:
        22:8f:98:b3:19:e0:20:e9:a1:32:ae:a6:ba:cc:b1:db:40:d3:
        8f:ed:8c:c2:41:20:db:ab:47:24:1f:03:fb:85:91:17:cf:50:
        41:50:97:d4:b3:48:8a:d3:6a:fd:21:4c:3e:52:70:ba:6d:a0:
        21:24:d2:f9:4b:ef:50:98:89:d8:27:45:50:90:70:a9:a4:9d:
        07:40:4c:53:47:b8:00:88:50:1a:5e:17:44:ff:aa:34:c0:29:
        02:f3
```

Finalmente en /etc/openvpn/server.conf definimos las directrices de configuración del servidor vpn, indicamos aquí las rutas a los certificados. Este fichero se generará dinámicamente a través de una plantilla de puppet.

Ficheros Anexos
/etc/puppet/modules/openvpn/manifests/init.pp
/etc/puppet/modules/openvpn/templates/openvpn.cfg

Conclusiones

Gracias a la consecución del proyecto se han confirmado los dos planteamientos iniciales, tanto la posibilidad de implantación para obtener un mayor control sobre la infraestructura actual, como la de ganar un argumento sólido de venta que potencie la línea emergente de negocio. Al realizar la puesta en producción ganaríamos flexibilidad y eficiencia a la hora de administrar los clientes existentes, aunque la migración de cada cliente supondrá un proyecto propio e individual.

Respecto a la herramienta utilizada la valoración es más que satisfactoria, originalmente Puppet está pensado para la administración de granjas de servidores en datacenters, sin embargo ha demostrado una gran flexibilidad para adaptarse a otro entorno completamente distinto sin ningún tipo de problema.

Ante el potencial que puede llegar a desplegar Puppet si llegásemos a desarrollar en Ruby sólo cabe decir que no ha sido explotado completamente en la consecución de este proyecto. Se debe profundizar en el desarrollo de plantillas y tipos en Ruby y de lentes de Augeas.

Cabe destacar que en cuanto a la planificación y dimensionado hubo errores que han provocado grandes desviaciones y han puesto en peligro la finalización del proyecto.

Inciden negativamente una serie de factores que analizaremos posteriormente. A pesar de que se han llegado a cumplir a tiempo todos los objetivos definidos en el alcance, la calidad final es inferior a la planteada en un primer momento.

Aunque el proyecto fue infradimensiado desde un primer momento, el factor más importante que ha jugado en contra ha sido la disponibilidad de recursos personales. No se ha cumplido la dedicación esperada durante la primera mitad del proyecto y sólo con una sobrecarga de esfuerzo y dedicación en la segunda mitad se han conseguido cumplir objetivos.

Si evaluamos objetivamente la desviación, como factor en contra también hemos de tener en cuenta los recursos de hardware utilizados. Puppet requiere de cantidad de memoria y ciclos de CPU para ejecutarse, tanto servidor maestro como el demonio de los clientes. Todo el trabajo de desarrollo se ha realizado en un portátil doméstico que cuando ejecuta las tres máquinas virtuales para simular la infraestructura propuesta ofrece un pésimo rendimiento. Los tiempos de espera para que el demonio descargue y aplique el catálogo durante el desarrollo han sido enormes y la depuración de errores en los mismos se ha eternizado.

Valorando también que no se ha tenido en cuenta a la hora de dimensionar el proyecto que se iba a trabajar con una tecnología desconocida, encontramos un factor de riesgo muy importante a la hora de estimar tiempo y esfuerzo necesarios.

Importante también a la hora de evaluar la desviación es tener en cuenta que se ha trabajado con software de terceros no disponible en los repositorios de la distribución elegida. Corregir errores en fuentes, compilar y generar los paquetes no estaba incluido en la planificación inicial del proyecto y ha consumido gran cantidad de tiempo.

Cuando se trata de seguridad en tecnologías de la información se puede hacer difícil distinguir entre precaución y paranoia. Al margen de esta cuestión se han abierto muchas líneas de trabajo para realizar mantenimientos evolutivos que mejoren la seguridad, capacidades y rendimiento de la infraestructura planteada.

Tras finalizar las plantillas aquí propuestas, otra línea abierta de trabajo sería continuar desarrollando plantillas para otros tipos de roles de servidores perimetrales. La base de trabajo se ha constituido con la personalización de la distribución CentOS, esta es la tarea más importante ya que a partir de aquí se pueden seguir añadiendo los módulos que sean necesarios para cumplir las funcionalidades requeridas.

Obviamente se podría plantear una ampliación de proyecto para realizar las tareas anteriormente mencionadas en lugar de reducirlo a mantenimientos evolutivos, ya que el volumen de las mismas es más que suficiente para justificar otro proyecto.

Respecto a la consecución final, podemos afirmar que disponemos de un punto centralizado de administración y mantenimiento de una red de servidores altamente escalable. Los servidores cumplen un perfil muy específico y cuentan con valores añadidos como monitorización, integridad, medidas de seguridad reactivas y proactivas, “DRP” y auditorías.

Bibliografía

- Puppet Labs Documentation
- Setting Up Puppet
- Getting Started with a Simple Puppet Pattern
- Using Puppet with Augeas
- Red Hat Puppet app
- Testing and Debugging your puppet configuration
- Puppet Certificates and Security
- Linux Kernel /etc/sysctl.conf Security Hardening
- Managing /etc/sysctl.conf with the sysctl utility
- AIDE - Advanced Intrusion Detection Environment
- Rootkit Hunter Wiki
- How To Install Rkhunter
- Install RKHunter
- Lynis Documentation
- Nagios Documentation
- Installing Nagios on CentOS 4.x/5.x
- Automatically build a nagios server and nrpe clients with puppet
- Using NRPE To Monitor Remote Services
- Munin Project Page
- Linux Help - *Portsentry* Setup Guide
- Sentry Tools
- Introducción a SELinux
- Implementación de SELinux
- MailScanner Documentation.
- ClamAV Antivirus
- Cómo configurar Clamd
- Postfix Basic Configuration.
- SpamAssassion Official Web
- OpenVPN Documentation

- Tony Bautts, Terry Dawson, Gregor N. Purdy. *LINUX Guía para administradores de redes.* Ed. O'Reilly
- Michael D. Bauer. *Seguridad en servidores LINUX*. Ed. O'Reilly.

- Jordi Blasco. *Materiales del Curso de Seguridad Informática Avanzada*. Universitat de Barcelona.

Anexos

Infraestructura de distribución de configuraciones

/etc/puppet/puppet.conf

```
[main]
    # The Puppet log directory.
    # The default value is '$vardir/log'.
    logdir = /var/log/puppet

    # Where Puppet PID files are kept.
    # The default value is '$vardir/run'.
    rundir = /var/run/puppet

    # Where SSL certificates are kept.
    # The default value is '$confdir/ssl'.
    ssldir = $vardir/ssl

[agent]
    # The file in which puppetd stores a list of the classes
    # associated with the retrieved configuratiion.  Can be loaded in
    # the separate ``puppet`` executable using the ``--loadclasses``
    # option.
    # The default value is '$confdir/classes.txt'.
    classfile = $vardir/classes.txt

    # Where puppetd caches the local configuration.  An
    # extension indicating the cache format is added automatically.
    # The default value is '$confdir/localconfig'.
    localconfig = $vardir/localconfig
```

Infraestructura de monitorización.

/etc/nagios/nagios.cfg

```
log_file=/var/log/nagios/nagios.log
cfg_file=/etc/nagios/objects/commands.cfg
cfg_file=/etc/nagios/objects/contacts.cfg
cfg_file=/etc/nagios/objects/timeperiods.cfg
cfg_file=/etc/nagios/objects/templates.cfg
cfg_file=/etc/nagios/objects/localhost.cfg
cfg_dir=/etc/nagios/conf.d
object_cache_file=/var/log/nagios/objects.cache
precached_object_file=/var/log/nagios/objects.precache
resource_file=/etc/nagios/private/resource.cfg
status_file=/var/log/nagios/status.dat
status_update_interval=10
nagios_user=nagios
nagios_group=nagios
check_external_commands=1
command_check_interval=-1
command_file=/var/spool/nagios/cmd/nagios.cmd
external_command_buffer_slots=4096
lock_file=/var/run/nagios.pid
temp_file=/var/log/nagios/nagios.tmp
temp_path=/tmp
event_broker_options=-1
log_rotation_method=d
log_archive_path=/var/log/nagios/archives
use_syslog=1
log_notifications=1
```

```
log_service_retries=1
log_host_retries=1
log_event_handlers=1
log_initial_states=0
log_external_commands=1
log_passive_checks=1
service_inter_check_delay_method=s
max_service_check_spread=30
service_interleave_factor=s
host_inter_check_delay_method=s
max_host_check_spread=30
max_concurrent_checks=0
check_result_reaper_frequency=10
max_check_result_reaper_time=30
check_result_path=/var/log/nagios/spool/checkresults
max_check_result_file_age=3600
cached_host_check_horizon=15
cached_service_check_horizon=15
enable_predictive_host_dependency_checks=1
enable_predictive_service_dependency_checks=1
soft_state_dependencies=0
auto_reschedule_checks=0
auto_rescheduling_interval=30
auto_rescheduling_window=180
sleep_time=0.25
service_check_timeout=60
host_check_timeout=30
event_handler_timeout=30
notification_timeout=30
ocsp_timeout=5
perfdata_timeout=5
retain_state_information=1
state_retention_file=/var/log/nagios/retention.dat
retention_update_interval=60
use_retained_program_state=1
use_retained_scheduling_info=1
retained_host_attribute_mask=0
retained_service_attribute_mask=0
retained_process_host_attribute_mask=0
retained_process_service_attribute_mask=0
retained_contact_host_attribute_mask=0
retained_contact_service_attribute_mask=0
interval_length=60
check_for_updates=1
bare_update_check=0
use_aggressive_host_checking=0
execute_service_checks=1
accept_passive_service_checks=1
execute_host_checks=1
accept_passive_host_checks=1
enable_notifications=1
enable_event_handlers=1
process_performance_data=0
obsess_over_services=0
obsess_over_hosts=0
translate_passive_host_checks=0
passive_host_checks_are_soft=0
check_for_orphaned_services=1
check_for_orphaned_hosts=1
check_service_freshness=1
service_freshness_check_interval=60
check_host_freshness=0
host_freshness_check_interval=60
additional_freshness_latency=15
enable_flap_detection=1
```

```
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=us
p1_file=/usr/sbin/p1.pl
enable_embedded_perl=1
use_embedded_perl_implicitly=1
illegal_object_name_chars=`~!$%^&*|'"<>?,()=
illegal_macro_output_chars=`~$&|'"<>
use_regexp_matching=0
use_true_regexp_matching=0
admin_email=nagios@localhost
admin_pager=pagenagios@localhost
daemon_dumps_core=0
use_large_installation_tweaks=0
enable_environment_macros=1
debug_level=16
debug_verbosity=1
debug_file=/var/log/nagios/nagios.debug
max_debug_file_size=1000000
```

Instalación del sistema base

/var/www/html/ks.cfg

```
# Kickstart file automatically generated by anaconda.

#version=RHEL6
install
cdrom
lang es_ES.UTF-8
keyboard es
network --device eth0 --bootproto dhcp
rootpw  --iscrypted
$6$o2tmMlDLAx.6wOHD$XdwwDPE5WrGd1yDjrfjSvUFDS19dQr0nIxH4oSGjRUQNoTCAT/Ym
Gs1m2onh3eV4RRo8nWQJiEhFi/k2iHb5W0

firewall --service=ssh
authconfig --enableshadow --passalgo=sha512 --enablefingerprint
selinux --enforcing
timezone --utc Europe/Madrid
bootloader --location=mbr --driveorder=sda --append="crashkernel=auto
crashkernel=auto rhgb quiet"
# The following is the partition information you requested
# Note that any partitions you deleted are not expressed
# here so unless you clear all partitions first, this is
# not guaranteed to work
#clearpart --none --drives=sda
#volgroup VolGroup --pesize=4096 pv.UAyadC-yirD-Hnfn-J3YA-5M5q-8vwD-
ZEb2VW
#logvol /home --fstype=ext4 --name=LogHome --vgname=VolGroup --size=200
#logvol / --fstype=ext4 --name=LogRoot --vgname=VolGroup --size=1500
#logvol swap --name=LogSwap --vgname=VolGroup --size=512
#logvol /tmp --fstype=ext4 --name=LogTmp --vgname=VolGroup --size=500
#logvol /var --fstype=ext4 --name=LogVar --vgname=VolGroup --size=600

#part /boot --fstype=ext4 --size=200
#part pv.UAyadC-yirD-Hnfn-J3YA-5M5q-8vwD-ZEb2VW --grow --size=3800

repo --name="centos"  --baseurl=file:///mnt/source --cost=100
repo --name="centos"  --baseurl=file:///mnt/source/ --cost=100

%packages --nobase
@Core
@core

%end
```

Optimización del Sistema Base

Módulo tunning

/etc/puppet/modules/tunning/manifests/init.pp

```
# init.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>

# See LICENSE for the full license granted to you.

# Main module class
#

class tunning{
	include tunning::ntp
	include tunning::ntp::nrpe
	include tunning::cron
	include tunning::disable_services
	include tunning::nrpe
	include tunning::repositories
	include tunning::utils
}
```

Clase tunning::ntp

/etc/puppet/modules/tunning/manifests/ntp.pp

```
# ntp.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs & configure ntpd
#
class tunning::ntp ($enable = true, $ensure = running) {
	$service_name = 'ntpd'
	$pkg_name = 'ntp'

	package { 'ntp':
		ensure	=>	installed,
	}

	service { 'ntpd':
		ensure =>	$ensure,
		name   =>	$service_name,
		enable	=>	$enable,
		subscribe => File['ntp.conf'],
	}

	file { 'ntp.conf':
		path   =>	'/etc/ntp.conf',
		ensure =>	file,
		require =>	Package['ntp'],
	}
}
```

/etc/puppet/modules/tunning/manifests/ntp/nrpe.pp

```
# nrpe.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

class tunning::ntp::nrpe{
```

```
        package{
                'nagios-plugins-ntp':
                        ensure =>     installed,
        }
        file{
                'nrpe.ntp.cfg':
                        ensure =>     file,
                        path   =>     "/etc/nrpe.d/nrpe.ntp.cfg",
                        content      =>
        "command[check_ntp_time]=/usr/lib/nagios/plugins/check_ntp_time
    -H 0.pool.ntp.org",
                        notify =>     Service['nrpe'],
        }
    }
```

Clase tunning::nrpe

/etc/puppet/modules/tunning/manifests/nrpe.pp

```
    # nrpe.pp
    # Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
    # See LICENSE for the full license granted to you.

    # Installs & configure nrpe
    #
    class tunning::nrpe ($enable = true, $ensure = running){
        file { 'nrpe.pref.cfg':
                        ensure =>     file,
                        path   =>     "/etc/nrpe.d/nrpe.pref.cfg",
                        require      =>     Package['nrpe'],
                        source =>
        [ "puppet:///modules/nrpe/nrpe.pref.cfg.$::fqdn",

    "puppet:///modules/nrpe/nrpe.pref.cfg"
                            ];
                'nrpe.general.commands.cfg':
                        ensure =>     file,
                        path         =>
        "/etc/nrpe.d/nrpe.general.commands.cfg",
                        content      =>
        template('nrpe/nrpe.server.erb'),
                        require      =>     Package['nrpe'],
                        source =>
        ["puppet:///modules/nrpe/nrpe.general.commands.cfg"],
        }

        service { 'nrpe':
                ensure       =>     $ensure,
                enable =>     $enable,
                subscribe => File['nrpe.pref.cfg',
                                  'nrpe.general.commands.cfg'],
                require      =>     Package['nrpe'],
        }
        $package_list = ["nrpe",
                "nagios-plugins-procs",
                "nagios-plugins-disk",
                "nagios-plugins-users",
                "nagios-plugins-load",
                "nagios-plugins-log",
        ]

        package{$package_list:    ensure => "installed", }
    }
```

/etc/puppet/modules/tunning/templates/nrpe.server.erb

```
allowed_hosts=<%= server %>
```

/etc/puppet/modules/tunning/files/nrpe.general.commands

```
command[check_disk_root]=/usr/lib/nagios/plugins/check_disk -w 20% -c
10% -p /dev/mapper/VolGroup-LogRoot
command[check_disk_home]=/usr/lib/nagios/plugins/check_disk -w 20% -c
10% -p /dev/mapper/VolGroup-LogHome
command[check_disk_tmp]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10%
-p /dev/mapper/VolGroup-LogTmp
command[check_disk_var]=/usr/lib/nagios/plugins/check_disk -w 20% -c 10%
-p /dev/mapper/VolGroup-LogVar
```

Clase tunning::munin

/etc/puppet/modules/tunning/manifests/munin.pp

```
# munin.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs & configure nrpe
#
class tunning::nrpe ($enable = false, $ensure = stopped){
	service { 'munin-node':
		ensure	=>	$ensure,
		enable =>	$enable,
		require	=>	Package['munin-node'],
	}
	$package_list = ["munin",
		"munin-node",
	]

	package{$package_list:	ensure => "installed", }
}
```

Clase tunning::utils

/etc/puppet/modules/tunning/manifests/utils.pp

```
# utils.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs utilities
#
class tunning::utils {
	$packages =  [ "htop",
			"tree",
			"man",
			"ccze",
			"mlocate",
			"file",
			]

	package {
		$packages:
			ensure	=>	installed,
	}
}
```

Clase tunning::cron

/etc/puppet/modules/tunning/manifests/cron.pp

```
# cron.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs service cron
#
class tunning::cron ($enable = true, $ensure = running) {
       package {
              'cronie':
                     ensure => installed,
       }

       service {
              'crond':
                     ensure => $ensure,
                     enable => $enable,
                     alias => 'crond',
                     hasstatus => true,
                     hasrestart => true,
       }
}
```

Clase tunning::repositories

/etc/puppet/modules/tunning/manifests/repostories.pp

```
# repositories.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs aditional yum repositories
#

class tunning::repositories{

       file {
              'repo.repo':
                     ensure => present,
                     source => "puppet:///modules/tunning/repo.repo",
                     path   => "/etc/yum.repos.d/repo.repo"
       }

}
```

/etc/puppet/modules/tunning/files/repo.repo

```
[repo.repo]
name=Repo Local
baseurl=http://master/repo/
enabled=1
gpgcheck=0
```

Clase tunning::disable_services

/etc/puppet/modules/tunning/manifests/disableservices.pp

```
# disable_services.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Disable unwanted services from booting up on startup
```

```
#
class tunning::disable_services {

       $service_list = ["rpcbind",
                        "rpcgssd",
                        "rpcidmapd",
                        "rpcsvcgssd",
                        "nfslock",
                        "fcoe",
                        ]

       service {
              $service_list:
                     ensure =>     "stopped",
                     enabled      =>     "false",
        }

}
```

Hardening del sistema base

Módulo se seguridad local

/etc/puppet/modules/local_security/manifests/init.pp

```
# local_security.pp - Local Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Main module class
#
class local_security{

       include local_security::password
       include local_security::defaultrunlevel
       include local_security::sysinit
       include local_security::term
       include local_security::extstorage
       include local_security::fmsount
       include local_security::issue
       include local_security::cronrestrictions
       include local_security::ctrlaltdel
       include local_security::ctrlaltdel::nrpe
}
```

Clase local_security::password

/etc/puppet/modules/local_security/manifests/password.pp

```
# password.pp - Local Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Class setting min password length
# TODO
#  - Augeas lense for this file
#  - Check password cypher<F2>

class local_security::password{

       file {
              'login.defs':
                     ensure =>     file,
                     path   =>     '/etc/login.defs',
                     source =>     [ "puppet:///modules/secure/login.defs.
$::fqdn" ,
                                   "puppet:///modules/secure/login.defs"
                                   ],
       }
}
```

/etc/puppet/modules/local_security/files/login.defs

```
MAIL_DIR     /var/spool/mail
PASS_MAX_DAYS99999
PASS_MIN_DAYS0
PASS_MIN_LEN 8
PASS_WARN_AGE7
UID_MIN                  500
UID_MAX                60000
GID_MIN                  500
```

```
GID_MAX                   60000
CREATE_HOME  yes
UMASK           077
USERGROUPS_ENAB yes
ENCRYPT_METHOD SHA512
```

Clase local_security::defaultrunlevel

/
etc/puppet/modules/local_security/manifests/defaultrunlevel

```
# local_security.pp - Local Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.


# Class to set default runlevel to 3
#
class local_security::defaultrunlevel{
      augeas { 'inittab':
            context      =>     '/files/etc/inittab',
            changes      =>     [
                  "set id/runlevels 3",
            ]
      }
}
```

Clase local_security::sysinit

/etc/puppet/modules/local_security/manifests/sysinit.pp

```
# sysinit.pp - Local Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Securing init process
#  Disable interactive startup
#  Ask password on single mode
#  Low active tty from 6 to 2
#
class local_security::sysinit{
      augeas {'init':
            context =>   '/files/etc/sysconfig/init',
            changes      => [
                  "set PROMPT no",
                  "set SINGLE /sbin/sulogin",
                  "set ACTIVE_CONSOLES /dev/tty[1-2]",
            ]
      }
}
```

Clase local_security::term

/etc/puppet/modules/local_security/manifests/term.pp

```
# term.pp - Local Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Securing terminal
#  Set active terminals timeout to 15 min
#  Set histfile readonly flag
class local_security::term{
```

```
        file {
                'os-security.sh':
                        path   =>     '/etc/profile.d/os-security.sh',
                        content      =>     "readonly TMOUT=900
                                            readonly HISTFILE",
                        mode   =>     644,
                        owner  =>     root,
                        growp  =>     root,
        }

}
```

Clase local_security::extstorage

/etc/puppet/modules/local_security/manifests/extstorage.pp

```
# extstorage.pp - Local Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.


# Disable usb and firewire local storage
#
class local_security::extstorage{

        file {
                'blacklist-usbstorage':
                        ensure =>     present,
                        path   =>     '/etc/modprobe.d/blacklist-
storage.conf',
                        content      =>     "blacklist usb-storage";

                'blacklist-firewire':
                        ensure =>     present,
                        path   =>     '/etc/modprobe.d/blacklist-
firewire.conf',
                        content      =>     "blacklist firewire_ohci";
        }

}
```

Clase local_security::fsmount

/etc/puppet/modules/local_security/manifiests/fsmount.pp

```
# fsmount.pp - Local Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.


# Secure filesystem mount options according FSH
#
class local_security::fsmount{

        augeas {
                'fstab':
                        context =>      '/files/etc/fstab',
                        changes => [
                                "set *[file = '/home']/opt
noexec,nodev,nosuid",
                                "set *[file = '/var']/opt
noexec,nodev,nosuid",
                                "set *[file = '/tmp']/opt
noexec,nodev,nosuid",
                        ],
```

```
        }
}
```

Clase local_security::issue

/etc/puppet/modules/local_security/manifests/issue.pp

```
# issue.pp - Local Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.


# Set legal messages on issue and issue.net
#
class local_security::issue{

        $files = ["/etc/issue","/etc/issue.net"]
        file {
                $files:
                        ensure =>     file,
                        source =>     [ "puppet:///modules/issue/issue.
$::fqdn",
                                        "puppet:///modules/issue/issue"
                                      ],
        }
}
```

/etc/puppet/modules/local_security/files/issue

```
                            ------------
                            W A R N I N G
                            ------------

THIS IS A PRIVATE COMPUTER SYSTEM.

This computer system including all related equipment, network devices
(specifically including Internet access), are provided only for
authorized use. All computer systems may be monitored for all lawful
purposes, including to ensure that their use is authorized, for
management of the system, to facilitate protection against unauthorized
access, and to verify security procedures, survivability and operational
security.
Monitoring includes active attacks by authorized personnel and their
entities to test or verify the security of the system. During
monitoring, information may be examined, recorded, copied and used for
authorized purposes. All information including personal information,
placed on or sent over this system may be monitored. Uses of this
system, authorized or unauthorized, constitutes consent to monitoring of
this system.
Unauthorized use may subject you to criminal prosecution. Evidence of
any such unauthorized use collected during monitoring may be used for
administrative, criminal or other adverse action. Use of this system
constitutes consent to monitoring for these purposes.
```

Clase local_security::cronrestrictions

/etc/puppet/modules/local_security/manifests/cronrestrictions.pp

```
# cronrestrictions.pp - Local Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.
```

```
# Restricts cron users to root
#
class local_security::cronrestrictions{

      exec {
            "cron.deny":
                  command => "awk -F: '{ print \$1 }' /etc/passwd |
grep -v root > /etc/cron.deny",
                   path    => [ "/bin/", "/usr/bin/" ],
      }

      file {
            "/etc/cron.deny":
                  mode    =>      600,
                   subscribe       =>      Exec['cron.deny'],
      }

      file {
            "/etc/cron.allow":
                  ensure  =>      file,
                  mode    =>      600,
                  content =>      "root",
      }
}
```

Clase local_security::ctrlaltdel

/etc/puppet/modules/local_security/manifests/ctrlaltdel.pp

```
# ctrlaltdel.pp - Local Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Disable ctrl+alt+delete reset combination key
# Log a warning to the syslog
# -TODO Augeas lense.
# -Nagios monit tool.
#
class local_security::ctrlaltdel{

      file {
            'control-alt-delete.conf':
                  ensure =>     file,
                  path   =>     '/etc/init/control-alt-delete.conf',
                  source =>     [ "puppet:///modules/secure/control-
alt-delete.conf.$::fqdn",
                                "puppet:///modules/secure/control-alt-
delete.conf"
                                ],
      }
}
```

/etc/puppet/modules/local_security/files/ctrl-alt-delete.conf

```
# control-alt-delete - emergency keypress handling
#
# This task is run whenever the Control-Alt-Delete key combination is
# pressed.  Usually used to shut down the machine.

start on control-alt-delete

exec logger -p WARN "Control-Alt-Delete pressed!!!"
```

/etc/puppet/modules/local_security/manifests/ctrlaltdel/nrpe.pp

```
# nrpe.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

class integrity_check::ctrlaltdel::nrpe{
       package {
              'nagios-check-log':
                     ensure => installed,
       }

       file{
              'nrpe.aide.cfg':
                     ensure =>     file,
                     path   =>     "/etc/nrpe.d/nrpe.ctrlaltdel.cfg",
                     content      =>
       "command[check_ctrlaltdel]=/usr/lib/nagios/plugins/check_log
-F /var/log/messages -O /tmp/messages.ctrlaltdel -q 'Control-Alt-Delete
pressed!!!"' ",
                     notify =>     Service['nrpe'];

              'messages':
                     ensure =>     file,
                     owner  =>     'root',
                     group  =>     'nrpe',
                     mode   =>     '0640',
                     path   =>     '/var/log/messages';
       }
}
```

Módulo de seguridad de red

/etc/puppet/modules/network_security/manifests/init.pp

```
# init.pp - Network Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Main module class
#
class network_security{

              include network_security::sysctl
              include network_security::portsentry
              include network_security::portsentry::nrpe
              include network_security::ssh
              include network_security::ssh::nrpe

}
```

Clase network_security::sysctl

/etc/puppet/modules/network_security/manifests/sysctl.pp

```
# sysctl.pp - Network Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Change sysctl parametres
#  Disables IP source routing
#  Disable ICMP Redirect Acceptance
```

```
#  Enable IP spoofing protection, turn on source route verification
#  Enable ignore broadcast request
#  Enable bad error message Protection
#  Log Spoofed Packets, Source routed packets
class network_security::sysctl{
      augeas {'sysctl':
            context =>   '/files/etc/sysctl.conf',
            changes      => [
                  "set net.ipv4.conf.all.rp_filter 1",
                  "set net.ipv4.conf.all.accept_source_route 0",
                  "set net.ipv4.conf.all.accept_redirects 0",
                  "set net.ipv4.tcp_syncookies 1",
                  "set net.ipv4.tcp_synact_retries 2",
                  "set net.ipv4.icmp_echo_ignore_broadcasts 1",
                  "set net.ipv4.icmp_ignore_bogus_error_responses 1",
                  "set net.ipv4.conf.all.log_martians 1",
                  "set kernel.sysrq 0",
            ]
      }

      exec {"sysctl":
              command => "sysctl -p",
            path    => [ "/sbin/" ],
            subscribe => Augeas['sysctl'],
      }
}
```

Clase network_security::portsentry

/root/porsentry.spec

```
%define version 1.2
%define release 1.rpm
%define name    portsentry

Name:        %{name}
Version:     %{version}
Release:     %{release}
Summary:     Hostlevel security protect against portscans.
Group:       System Environment/Daemons
License:   CPL
Packager:    Sergio Pena <kekio.one@gmail.com>
URL:         http://sentrytools.sourceforge.net

%description
The Sentry tools provide host-level security services for the Unix
platform. Por
tSentry, Logcheck/LogSentry, and HostSentry protect against portscans,
automate
log file auditing, and detect suspicious login activity on a continuous
basis.

%clean
rm -rf ${RPM_BUILD_ROOT}


%post
# Sort out the rc.d directories
chkconfig --add portsentry
chkconfig portsentry on


%files
%defattr (644,root,root)
%attr(700,root,root) /usr/bin/portsentry
```

```
%attr(700,root,root) /etc/portsentry/portsentry.conf
%config(noreplace) %attr(700,root,root) /etc/portsentry/portsentry.conf
%config(noreplace) %attr(755,root,root) /%
{_sysconfdir}/rc.d/init.d/portsentry
```

/etc/portsentry/portsentry.conf

```
TCP_PORTS="1,11,15,79,111,119,143,540,635,1080,1524,2000,5742,6667,12345
,12346,20034,27665,31337,32771,32772,32773,32774,40421,49724,54320"
UDP_PORTS="1,7,9,69,161,162,513,635,640,641,700,37444,34555,31335,32770,
32771,32772,32773,32774,31337,54321"
ADVANCED_PORTS_TCP="1024"
ADVANCED_PORTS_UDP="1024"
ADVANCED_EXCLUDE_TCP="113,139"
ADVANCED_EXCLUDE_UDP="520,138,137,67"
IGNORE_FILE="/var/spool/portsentry/portsentry.ignore""
HISTORY_FILE="/var/spool/portsentry/portsentry.history"
BLOCKED_FILE="/var/spool/portsentry/portsentry.blocked"
RESOLVE_HOST = "1"
BLOCK_UDP="1"
BLOCK_TCP="1"
KILL_ROUTE="iptables -A INPUT -s $TARGET$ -j DROP"
KILL_HOSTS_DENY="ALL: $TARGET$"
SCAN_TRIGGER="0"
```

/etc/rc.d/init.d/portsentry

```
#!/bin/sh
#
# Startup script for the PortSentry
#
# chkconfig: 2345 20 39
# description: Run two instances of portsentry
# processname: portsentry
# pidfile: /var/run/portsentry/portsentry.pid
# config: /etc/portsentry/portsentry.conf

# Source function library.
. /etc/rc.d/init.d/functions

[ -x /usr/bin/portsentry ] || exit 0

# See how we were called.
case "$1" in
  start)
       echo -n "Starting PortSentry Listeners"
       daemon portsentry -tcp
       RETVAL=$?
       echo
       [ $RETVAL -eq 0 ] && touch /var/lock/subsys/portsentry
       ;;
  stop)
       echo -n "Stopping PortSentry Listeners"
       killproc portsentry
       rm -f /var/run/portsentry/portsentry.pid
       RETVAL=$?
       echo
### heres the fix... we gotta remove the stale files on restart
       [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/portsentry
       ;;
  status)
       status portsentry
       RETVAL=$?
       ;;
  restart|reload)
       $0 stop
```

```
        $0 start
        RETVAL=$?
        ;;
  *)
        echo "Usage: portsentry {start|stop|status|restart|reload}"
        exit 1
esac

exit $RETVAL
```

/etc/puppet/modules/network_security/manifests/portsentry.pp

```
# portsentry.pp - Network Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs portsentry
#
class network_security::portsentry {

        service {
                'portsentry':
                        name   =>     'portsentry',
                        ensure =>     running,
                        enable       =>     true,
                        require       =>
        File['/etc/portsentry/portsentry.conf'],
        }

        package {
                'portsentry':
                        ensure =>     installed,
        }
}
```

/etc/puppet/modules/network_security/manifests/portsentry/nrpe.pp

```
# nrpe.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

class network_security::portsentry::nrpe{
        package {
                'nagios-check-log':
                        ensure => installed,
        }

        file{
                'nrpe.portsentry.cfg':
                        ensure =>     file,
                        path   =>     "/etc/nrpe.d/nrpe.ctrlaltdel.cfg",
                        content      =>
        "command[check_portsentry]=/usr/lib/nagios/plugins/check_log
-F /var/log/messages -O /tmp/messages.portsentry.old -q 'attackalert'",
                        notify =>     Service['nrpe'];

                'messages':
                        ensure =>     file,
                        owner  =>     'root',
                        group  =>     'nrpe',
                        mode   =>     '0640',
                        path   =>     '/var/log/messages';
```

```
        }
}
```

Clase network_security::ssh

/etc/puppet/modules/network_security/ssh.pp

```
# ssh.pp - Network Security enhancements
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Change ssh configuration
#
class ssh {

        package { 'openssh-server':
                ensure => installed,
        }

        service { 'sshd':
                ensure       =>     running,
                name   =>     'sshd',
                enable =>     true,
                subscribe => File['sshd_config'],
        }

        file { 'sshd_config':
                path   =>     '/etc/ssh/sshd_config',
                ensure =>     file,
                require      =>     Package['openssh-server'],
                source =>
        "puppet:///modules/network_security/sshd_config",
        }

}
```

/etc/puppet/modules/network_security/ssh/nrpe.pp

```
# nrpe.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

class network_security::ssh::nrpe{
        package {
                'nagios-check-log':
                        ensure => installed,
        }

        file{
                'nrpe.ssh.cfg':
                        ensure =>     file,
                        path   =>     "/etc/nrpe.d/nrpe.ssh.cfg",
                        content      =>
        "command[check_ssh_log]=/usr/lib/nagios/plugins/check_log -F
/var/log/secure -O /tmp/secure.ssh.old -q 'Failed password for'",
                        notify =>     Service['nrpe'];

                'messages':
                        ensure =>     file,
                        owner  =>     'root',
                        group  =>     'nrpe',
                        mode   =>     '0640',
                        path   =>     '/var/log/secure';
        }
}
```

Clase network_security::iptables

/etc/puppet/modules/network_security/manifests/iptables.pp

```
# iptables.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.


# Iptables managemente
class iptables {
	file { 'iptables':
		path   =>     "/etc/sysconfig/iptables",
		ensure =>     file,
		source =>     [ "puppet:///modules/iptables/iptables.
$::fqdn",
		                "puppet:///modules/iptables/iptables"
		              ],
		}
	service { 'iptables':
		name   =>     $service_name,
		ensure       =>     running,
		enable =>     true,
		subscribe => File['iptables'],
	}
}
```

/etc/puppet/modules/network_security/files/iptables

```
# Firewall configuration written by system-config-firewall
# Manual customization of this file is not recommended.
*filter
:INPUT DROP [0:0]
:FORWARD DROP [0:0]
:OUTPUT DROP [0:0]
-A INPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A INPUT -m state --state INVALID -j DROP
-A INPUT -p icmp -j ACCEPT
-A INPUT -i lo -j ACCEPT
-A INPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A INPUT -j REJECT --reject-with icmp-host-prohibited
-A OUTPUT -m state --state ESTABLISHED,RELATED -j ACCEPT
-A OUTPUT -m state --state NEW -m tcp -p tcp --dport 80 -j ACCEPT
-A OUTPUT -m state --state NEW -m udp -p udp --dport 53 -j ACCEPT
-A OUTPUT -m state --state NEW -m tcp -p tcp --dport 22 -j ACCEPT
-A FORWARD -j REJECT --reject-with icmp-host-prohibited
COMMIT
```

Módulo de comprobación de integridad

/etc/puppet/modules/integrity_check/manifests/init.pp

```
# integrity_check.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.


# Main module class
#
class integrity_check{

	include integrity_check::aide
```

```
        include      integrity_check::aide::nrpe

        include integrity_check::rkhunter
        include integrity_check::rkhunter::nrpe

        include integrity_check::lynis

}
```

Clase integrity_check::aide

/etc/puppet/modules/integrity_check/manifests/aide.pp

```
# aide.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.


# Installs & configure checks from aide
#
class integrity_check::aide{

        package{ 'aide':
                ensure => installed,
                require => Class["utils::cron"],
        }

        file{
                'aide.db':
                        ensure =>     file,
                        path   =>     "/var/lib/aide/aide.db.gz",
                        source =>     [
                                      "puppet:///modules/integrity_check/aide
.db.gz.$::fqdn",
                                      "puppet:///modules/integrity_check/aide
.db",
                                      ],
                        owner  =>     root,
                        group  =>     root,
                        mode   =>     600;

                'aide_db_sync':
                        ensure =>     file,
                        path   =>     "/usr/sbin/aide_db_sync",
                        source =>
        [ "puppet:///modules/integrity_check/aide_db_sync" ],
                        owner  =>     root,
                        group  =>     root,
                        mode   =>     700,
        }

        cron{
                'aide':
                        command      =>     '/usr/sbin/aide --check',
                        user   =>     root,
                        hour   =>     1,
                        minute =>     0,
                        require      =>     Class['utils::cron'],
        }

}
```

/etc/puppet/modules/integrity_check/manifests/aide/nrpe.pp

```
# nrpe.pp
```

```
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

class integrity_check::aide::nrpe{

	file{
		'nrpe.aide.cfg':
			ensure =>	file,
			path   =>	"/etc/nrpe.d/nrpe.aide.cfg",
			content	=>
	"command[check_aide]=/usr/lib/nagios/plugins/check_log -F
/var/log/aide/aide.log -O /tmp/aide.old -q 'AIDE found differences
between database and filesystem!!' ",
			notify =>	Service['nrpe'];

		'aide':
			ensure =>	file,
			owner  =>	'root',
			group  =>	'nrpe',
			mode   =>	'0640',
			path   =>	'/var/log/aide/aide.log',
	}
}
```

/etc/puppet/modules/integrity_check/files/aide_db_sync

```
#!/bin/bash
if [ -f /var/lib/aide/aide.db.new.gz ]
then
	scp /var/lib/aide/aide.db.new.gz
master:/etc/puppet/modules/integrity_check/files/aide.db.gz.$(hostname)
fi
```

Clase integrity_check::rkhunter

/etc/puppet/modules/integrity_check/manifests/rkhunter.pp

```
# aide.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.


# Installs & configure checks from rkhunter
#
class integrity_check::rkhunter{

	package{ 'rkhunter':
		ensure => installed,
		require => Class["utils::cron"],
	}

	cron{
		'rkhunter':
			command	=>	'/usr/bin/rkhunter --update
&& /usr/bin/rkhunter -c -rwo --cronjob 2>&1',
			user   =>	root,
			weekday	=>	1,
			hour   =>	1,
			minute =>	0,
			require	=>	Class['utils::cron'],
	}

}
```

/

/etc/puppet/modules/integrity_check/manifests/rkhunter/nrpe.pp

```
# nrpe.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

class integrity_check::rkhunter::nrpe{

	file{
		'nrpe.rkhunter.cfg':
			ensure =>	file,
			path   =>	"/etc/nrpe.d/nrpe.rkhunter.cfg",
			content	=>
	"command[check_rkhunter]=/usr/lib/nagios/plugins/check_log -F
/var/log/rkhunter/rkhunter.log -O /tmp/rkhunter.old -q 'WARNING'",
			notify =>	Service['nrpe'];

		'rkhunter':
			ensure =>	file,
			owner  =>	'root',
			group  =>	'nrpe',
			mode   =>	'0640',
			path   =>	'/var/log/rkhunter/rkhunter.log',
	}
}
```

Clase integrity_check::lynis

/etc/puppet/modules/integrity_check/manifests/lynis.pp

```
# lynis.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.


# Installs & configure lynis security auditor
#
class integrity_check::lynis{

	package{ 'lynis':
		ensure => installed,
		require => Class["tunning::cron"],
	}

	cron{
		'lynis':
			command	=>	'/usr/bin/lynis -c --quiet --no-
colors',
			user   =>	root,
			hour   =>	1,
			minute =>	0,
			require	=>	Class['tunning::cron'],
	}

}
```

/etc/puppet/modules/integrity_check/manifests/lynis/nrpe.pp

```
# nrpe.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

class integrity_check::lynis::nrpe{
```

```
	file{
		'nrpe.lynis.cfg':
			ensure =>	file,
			path   =>	"/etc/nrpe.d/nrpe.lynis.cfg",
			content	=>
	"command[check_lynis]=/usr/lib/nagios/plugins/check_log -F
/var/log/lynis.log -O /tmp/lynis.log -q 'WARNING'",
			notify =>	Service['nrpe'];

		'aide':
			ensure =>	file,
			owner  =>	'root',
			group  =>	'nrpe',
			mode   =>	'0640',
			path   =>	'/var/log/lynis.log',
	}
}
```

Plantilla de servidor AntiSpam

/etc/puppet/modules/aspam/manifests/init.pp

```
class aspam {
      include aspam::clamd
      include aspam::ms
      include aspam::postfix
      include aspam::spamassassin
}
```

Clase aspam::postfix

/etc/puppet/modules/aspam/manifests/postfix.pp

```
# postfix.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs & configure postfix
#     Modify header_checks to hold all incoming messages
#           for MailScanner
#     Configure mta parametres
#     Configure smtpd restrictions an rbl
#

class aspam::postfix ($myhostname = $::fqdn, $mydomain = empty,
$relaydomains = empty, $relayhost = empty, $mynetworks = empty,
$smtpd_banner = "$::fqdn SMTP Mail Server") {

      package {
                'postfix':
                        ensure  =>      'installed',
       }

      service {'postfix': enable => false}

       augeas {
                'main.cf':
                    require      =>     Package["postfix"],
                        context =>      '/files/etc/postfix/main.cf',
                        changes =>      [
                                "set myhostname $myhostname",
                          "set header_checks
regexp:/etc/postfix/header_checks",
                                "set mydomain $mydomain",
                                "set relaydomains $relaydomains",
                                "set relayhost  $relayhost",
                                "set mynetworks $mynetworks",
                                "set smtpd_banner $smtpd_banner",
                                "set disable_vrfy_command yes",
                                "set smtdp_helo_required yes",
                                "set smtpd_helo_restrictions
'permit_mynetworks,
        reject_invalid_hostname,
        reject_non_fqdn_hostname,
        permit'",

                                "set smtpd_recipient_restrictions
'permit_mynetworks,
        reject_invalid_hostname,
       reject_non_fqdn_hostname,
        reject_rbl_client zen.spamhaus.org,
```

```
        permit'",

                        ],
        }


      file {
            'header_checks':
                  ensure =>     present,
                  path   =>     '/etc/postfix/header_checks',
                  content      =>     '/^Received:/ HOLD',
                  require      =>     Package["postfix"],
      }
}
```

Clase aspam::ms

/etc/puppet/modules/aspam/manifests/ms.pp

```
# mailscanner.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs & configure mailscanner
#

class aspam::ms{
      $package_list = ["mailscanner",
            "perl-DBD-SQLite",
            "perl-Net-CIDR",
             "perl-Time-HiRes",
            "perl-Filesys-Df",
             "perl-Sys-Hostname-Long",
                  "perl-DBI",
             "perl-Sys-SigAction",
                  "perl-HTML-TokeParser-Simple",
            "perl-Archive-Zip",
             "perl-OLE-Storage_Lite",
             "perl-Compress-Zlib",
      ]

      package{$package_list:    ensure => "installed", }

        service{'MailScanner':
                        ensure  =>      running,
                        enable  =>      true,
                  subscribe    =>     File['MailScanner.conf'],
                  require      =>     Package[$package_list],
        }

        file {'MailScanner.conf':
                path    =>      '/etc/MailScanner/MailScanner.conf',
                ensure  =>      file,
                source  =>
[ "puppet:///modules/aspam/MailScanner.conf.$::fqdn" ],
            notify =>     Service['MailScanner'],
        }
        file {'MailScanner.prefs.conf':
                path    =>
'/etc/MailScanner/conf.d/MailScanner.prefs.conf',
                ensure  =>      file,
                source  =>
[ "puppet:///modules/aspam/MailScanner.prefs.conf" ],
        }
}
```

```
%org-name% = yoursite
%org-long-name% = Your Organisation Name Here
%web-site% = www.your-organisation.com
%etc-dir% = /etc/MailScanner
%report-dir% = /etc/MailScanner/reports/en
%rules-dir% = /etc/MailScanner/rules
%mcp-dir% = /etc/MailScanner/mcp
Max Children = 5
Run As User =
Run As Group =
Queue Scan Interval = 6
Incoming Queue Dir = /var/spool/mqueue.in
Outgoing Queue Dir = /var/spool/mqueue
Incoming Work Dir = /var/spool/MailScanner/incoming
Quarantine Dir = /var/spool/MailScanner/quarantine
PID file = /var/run/MailScanner.pid
Restart Every = 7200
MTA = sendmail
Sendmail = /usr/sbin/sendmail
Sendmail2 = /usr/sbin/sendmail
Incoming Work User =
Incoming Work Group =
Incoming Work Permissions = 0600
Quarantine User =
Quarantine Group =
Quarantine Permissions = 0600
Max Unscanned Bytes Per Scan = 100m
Max Unsafe Bytes Per Scan = 50m
Max Unscanned Messages Per Scan = 30
Max Unsafe Messages Per Scan = 30
Max Normal Queue Size = 800
Scan Messages = yes
Reject Message = no
Maximum Processing Attempts = 6
Processing Attempts Database =
/var/spool/MailScanner/incoming/Processing.db
Maximum Attachments Per Message = 200
Expand TNEF = yes
Use TNEF Contents = replace
Deliver Unparsable TNEF = no
TNEF Expander = /usr/bin/tnef --maxsize=100000000
TNEF Timeout = 120
File Command = /usr/bin/file
File Timeout = 20
Gunzip Command = /bin/gunzip
Gunzip Timeout = 50
Unrar Command = /usr/bin/unrar
Unrar Timeout = 50
Find UU-Encoded Files = no
Maximum Message Size = %rules-dir%/max.message.size.rules
Maximum Attachment Size = -1
Minimum Attachment Size = -1
Maximum Archive Depth = 8
Find Archives By Content = yes
Unpack Microsoft Documents = yes
Zip Attachments = no
Attachments Zip Filename = MessageAttachments.zip
Attachments Min Total Size To Zip = 100k
Attachment Extensions Not To Zip = .zip .rar .gz .tgz .jpg .jpeg .mpg
.mpe .mpeg .mp3 .rpm .htm .html .eml
Add Text Of Doc = no
Antiword = /usr/bin/antiword -f
Antiword Timeout = 50
Unzip Maximum Files Per Archive = 0
```

```
Unzip Maximum File Size = 50k
Unzip Filenames = *.txt *.ini *.log *.csv
Unzip MimeType = text/plain
Virus Scanning = yes
Virus Scanners = auto
Virus Scanner Timeout = 300
Deliver Disinfected Files = no
Silent Viruses = HTML-IFrame All-Viruses
Still Deliver Silent Viruses = no
Non-Forging Viruses = Joke/ OF97/ WM97/ W97M/ eicar
Spam-Virus Header = X-%org-name%-MailScanner-SpamVirus-Report:
Virus Names Which Are Spam = Sane*UNOFFICIAL HTML/* *Phish*
Block Encrypted Messages = no
Block Unencrypted Messages = no
Allow Password-Protected Archives = no
Check Filenames In Password-Protected Archives = yes
Allowed Sophos Error Messages =
Sophos IDE Dir = /opt/sophos-av/lib/sav
Sophos Lib Dir = /opt/sophos-av/lib
Monitors For Sophos Updates = /opt/sophos-av/lib/sav/*.ide
Monitors for ClamAV Updates = /usr/local/share/clamav/*.cld
/usr/local/share/clamav/*.cvd
ClamAVmodule Maximum Recursion Level = 8
ClamAVmodule Maximum Files = 1000
ClamAVmodule Maximum File Size = 10000000 # (10 Mbytes)
ClamAVmodule Maximum Compression Ratio = 250
Clamd Port = 3310
Clamd Socket = /tmp/clamd.socket
Clamd Lock File = # /var/lock/subsys/clamd
Clamd Use Threads = no
ClamAV Full Message Scan = yes
Fpscand Port = 10200
Dangerous Content Scanning = yes
Allow Partial Messages = no
Allow External Message Bodies = no
Find Phishing Fraud = yes
Also Find Numeric Phishing = yes
Use Stricter Phishing Net = yes
Highlight Phishing Fraud = yes
Phishing Safe Sites File = %etc-dir%/phishing.safe.sites.conf
Phishing Bad Sites File = %etc-dir%/phishing.bad.sites.conf
Country Sub-Domains List = %etc-dir%/country.domains.conf
Allow IFrame Tags = disarm
Allow Form Tags = disarm
Allow Script Tags = disarm
Allow WebBugs = disarm
Ignored Web Bug Filenames = spacer pixel.gif pixel.png gap shim
Known Web Bug Servers = msgtag.com
Web Bug Replacement = http://www.mailscanner.tv/1x1spacer.gif
Allow Object Codebase Tags = disarm
Convert Dangerous HTML To Text = no
Convert HTML To Text = no
Archives Are = zip rar ole
Allow Filenames =
Deny Filenames =
Filename Rules = %etc-dir%/filename.rules.conf
Allow Filetypes =
Allow File MIME Types =
Deny Filetypes =
Deny File MIME Types =
Filetype Rules = %etc-dir%/filetype.rules.conf
Archives: Allow Filenames =
Archives: Deny Filenames =
Archives: Filename Rules = %etc-dir%/archives.filename.rules.conf
Archives: Allow Filetypes =
```

```
Archives: Allow File MIME Types =
Archives: Deny Filetypes =
Archives: Deny File MIME Types =
Archives: Filetype Rules = %etc-dir%/archives.filetype.rules.conf
Default Rename Pattern = __FILENAME__.disarmed
Quarantine Infections = yes
Quarantine Silent Viruses = no
Quarantine Modified Body = no
Quarantine Whole Message = no
Quarantine Whole Messages As Queue Files = no
Keep Spam And MCP Archive Clean = no
Language Strings = %report-dir%/languages.conf
Rejection Report = %report-dir%/rejection.report.txt
Deleted Bad Content Message Report  = %report-dir
%/deleted.content.message.txt
Deleted Bad Filename Message Report = %report-dir
%/deleted.filename.message.txt
Deleted Virus Message Report        = %report-dir
%/deleted.virus.message.txt
Deleted Size Message Report        = %report-dir
%/deleted.size.message.txt
Stored Bad Content Message Report  = %report-dir
%/stored.content.message.txt
Stored Bad Filename Message Report = %report-dir
%/stored.filename.message.txt
Stored Virus Message Report        = %report-dir
%/stored.virus.message.txt
Stored Size Message Report        = %report-dir%/stored.size.message.txt
Disinfected Report = %report-dir%/disinfected.report.txt
Inline HTML Signature = %report-dir%/inline.sig.html
Inline Text Signature = %report-dir%/inline.sig.txt
Signature Image Filename = %report-dir%/sig.jpg
Signature Image <img> Filename = signature.jpg
Inline HTML Warning = %report-dir%/inline.warning.html
Inline Text Warning = %report-dir%/inline.warning.txt
Sender Content Report        = %report-dir%/sender.content.report.txt
Sender Error Report        = %report-dir%/sender.error.report.txt
Sender Bad Filename Report = %report-dir%/sender.filename.report.txt
Sender Virus Report        = %report-dir%/sender.virus.report.txt
Sender Size Report         = %report-dir%/sender.size.report.txt
Hide Incoming Work Dir = yes
Include Scanner Name In Reports = yes
Mail Header = X-%org-name%-MailScanner:
Spam Header = X-%org-name%-MailScanner-SpamCheck:
Spam Score Header = X-%org-name%-MailScanner-SpamScore:
Information Header = X-%org-name%-MailScanner-Information:
Add Envelope From Header = yes
Add Envelope To Header = no
Envelope From Header = X-%org-name%-MailScanner-From:
Envelope To Header = X-%org-name%-MailScanner-To:
ID Header = X-%org-name%-MailScanner-ID:
IP Protocol Version Header = # X-%org-name%-MailScanner-IP-Protocol:
Spam Score Character = s
SpamScore Number Instead Of Stars = no
Minimum Stars If On Spam List = 0
Clean Header Value       = Found to be clean
Infected Header Value    = Found to be infected
Disinfected Header Value = Disinfected
Information Header Value = Please contact the ISP for more information
Detailed Spam Report = yes
Include Scores In SpamAssassin Report = yes
Always Include SpamAssassin Report = no
Multiple Headers = append
Place New Headers At Top Of Message = no
Hostname = the %org-name% ($HOSTNAME) MailScanner
```

```
Sign Messages Already Processed = no
Sign Clean Messages = yes
Attach Image To Signature = no
Attach Image To HTML Message Only = yes
Allow Multiple HTML Signatures = no
Dont Sign HTML If Headers Exist = # In-Reply-To: References:
Mark Infected Messages = yes
Mark Unscanned Messages = yes
Unscanned Header Value = Not scanned: please contact your Internet E-
Mail Service Provider for details
Remove These Headers = X-Mozilla-Status: X-Mozilla-Status2:
Deliver Cleaned Messages = yes
Notify Senders = yes
Notify Senders Of Viruses = no
Notify Senders Of Blocked Filenames Or Filetypes = yes
Notify Senders Of Blocked Size Attachments = no
Notify Senders Of Other Blocked Content = yes
Never Notify Senders Of Precedence = list bulk
Scanned Modify Subject = no # end
Scanned Subject Text = {Scanned}
Virus Modify Subject = start
Virus Subject Text = {Virus?}
Filename Modify Subject = start
Filename Subject Text = {Filename?}
Content Modify Subject = start
Content Subject Text = {Dangerous Content?}
Size Modify Subject = start
Size Subject Text = {Size}
Disarmed Modify Subject = start
Disarmed Subject Text = {Disarmed}
Phishing Modify Subject = no
Phishing Subject Text = {Fraud?}
Spam Modify Subject = start
Spam Subject Text = {Spam?}
High Scoring Spam Modify Subject = start
High Scoring Spam Subject Text = {Spam?}
Warning Is Attachment = yes
Attachment Warning Filename = %org-name%-Attachment-Warning.txt
Attachment Encoding Charset = ISO-8859-1
Archive Mail =
Missing Mail Archive Is = directory
Send Notices = yes
Notices Include Full Headers = yes
Hide Incoming Work Dir in Notices = no
Notice Signature = -- \nMailScanner\nEmail Virus
Scanner\nwww.mailscanner.info
Notices From = MailScanner
Notices To = postmaster
Local Postmaster = postmaster
Spam List Definitions = %etc-dir%/spam.lists.conf
Virus Scanner Definitions = %etc-dir%/virus.scanners.conf
Spam Checks = yes
Spam List = # spamhaus-ZEN # You can un-comment this to enable them
Spam Domain List =
Spam Lists To Be Spam = 1
Spam Lists To Reach High Score = 3
Spam List Timeout = 10
Max Spam List Timeouts = 7
Spam List Timeouts History = 10
Is Definitely Not Spam = %rules-dir%/spam.whitelist.rules
Is Definitely Spam = no
Definite Spam Is High Scoring = no
Ignore Spam Whitelist If Recipients Exceed = 20
Max Spam Check Size = 200k
Use Watermarking = no
```

```
Add Watermark = yes
Check Watermarks With No Sender = yes
Treat Invalid Watermarks With No Sender as Spam = nothing
Check Watermarks To Skip Spam Checks = yes
Watermark Secret = %org-name%-Secret
Watermark Lifetime = 604800
Watermark Header = X-%org-name%-MailScanner-Watermark:
Use SpamAssassin = yes
Max SpamAssassin Size = 200k
Required SpamAssassin Score = 6
High SpamAssassin Score = 10
SpamAssassin Auto Whitelist = yes
SpamAssassin Timeout = 75
Max SpamAssassin Timeouts = 10
SpamAssassin Timeouts History = 30
Check SpamAssassin If On Spam List = yes
Include Binary Attachments In SpamAssassin = no
Spam Score = yes
Cache SpamAssassin Results = yes
SpamAssassin Cache Database File =
/var/spool/MailScanner/incoming/SpamAssassin.cache.db
Rebuild Bayes Every = 0
Wait During Bayes Rebuild = no
Use Custom Spam Scanner = no
Max Custom Spam Scanner Size = 20k
Custom Spam Scanner Timeout = 20
Max Custom Spam Scanner Timeouts = 10
Custom Spam Scanner Timeout History = 20
Spam Actions = deliver header "X-Spam-Status: Yes"
High Scoring Spam Actions = store
Non Spam Actions = deliver header "X-Spam-Status: No"
SpamAssassin Rule Actions =
Sender Spam Report         = %report-dir%/sender.spam.report.txt
Sender Spam List Report    = %report-dir%/sender.spam.rbl.report.txt
Sender SpamAssassin Report = %report-dir%/sender.spam.sa.report.txt
Inline Spam Warning = %report-dir%/inline.spam.warning.txt
Recipient Spam Report = %report-dir%/recipient.spam.report.txt
Enable Spam Bounce = %rules-dir%/bounce.rules
Bounce Spam As Attachment = no
Syslog Facility = mail
Log Speed = no
Log Spam = no
Log Non Spam = no
Log Delivery And Non-Delivery = no
Log Permitted Filenames = no
Log Permitted Filetypes = no
Log Permitted File MIME Types = no
Log Silent Viruses = no
Log Dangerous HTML Tags = no
Log SpamAssassin Rule Actions = yes
SpamAssassin Temporary Dir =
/var/spool/MailScanner/incoming/SpamAssassin-Temp
SpamAssassin User State Dir =
SpamAssassin Install Prefix =
SpamAssassin Site Rules Dir = /etc/mail/spamassassin
SpamAssassin Local Rules Dir =
SpamAssassin Local State Dir = # /var/lib/spamassassin
SpamAssassin Default Rules Dir =
DB DSN =
DB Username =
DB Password =
SQL Serial Number =
SQL Quick Peek =
SQL Config =
SQL Ruleset =
```

```
SQL SpamAssassin Config =
SQL Debug = no
MCP Checks = no
First Check = spam
MCP Required SpamAssassin Score = 1
MCP High SpamAssassin Score = 10
MCP Error Score = 1
MCP Header = X-%org-name%-MailScanner-MCPCheck:
Non MCP Actions = deliver
MCP Actions = deliver
High Scoring MCP Actions = deliver
Bounce MCP As Attachment = no
MCP Modify Subject = start
MCP Subject Text = {MCP?}
High Scoring MCP Modify Subject = start
High Scoring MCP Subject Text = {MCP?}
Is Definitely MCP = no
Is Definitely Not MCP = no
Definite MCP Is High Scoring = no
Always Include MCP Report = no
Detailed MCP Report = yes
Include Scores In MCP Report = no
Log MCP = no
MCP Max SpamAssassin Timeouts = 20
MCP Max SpamAssassin Size = 100k
MCP SpamAssassin Timeout = 10
MCP SpamAssassin Prefs File = %mcp-dir%/mcp.spam.assassin.prefs.conf
MCP SpamAssassin User State Dir =
MCP SpamAssassin Local Rules Dir = %mcp-dir%
MCP SpamAssassin Default Rules Dir = %mcp-dir%
MCP SpamAssassin Install Prefix = %mcp-dir%
Recipient MCP Report = %report-dir%/recipient.mcp.report.txt
Sender MCP Report = %report-dir%/sender.mcp.report.txt
Use Default Rules With Multiple Recipients = no
Read IP Address From Received Header = no
Spam Score Number Format = %d
MailScanner Version Number = 4.84.3
SpamAssassin Cache Timings = 1800,300,10800,172800,600
Debug = no
Debug SpamAssassin = no
Run In Foreground = no
Always Looked Up Last = no
Always Looked Up Last After Batch = no
Deliver In Background = yes
Delivery Method = batch
Split Exim Spool = no
Lockfile Dir = /var/spool/MailScanner/incoming/Locks
Custom Functions Dir = /usr/lib/MailScanner/MailScanner/CustomFunctions
Lock Type =
Syslog Socket Type =
Automatic Syntax Check = yes
Minimum Code Status = supported
include /etc/MailScanner/conf.d/*
```

/etc/puppet/modules/aspam/files/MailScanner.prefs.conf

```
Run As User = postfix
Run As Group = postfix
Incoming Queue Dir = /var/spool/postfix/hold
Outgoing Queue Dir = /var/spool/postfix/incoming
MTA = postfix
Incoming Work User =
Incoming Work Group = clam
Incoming Work Permissions = 0660
Quarantine User = root
Quarantine Group = apache
```

```
Quarantine Permissions = 0660
Virus Scanners = clamd
Clamd Socket = /var/run/clamav/clamd.sock
Quarantine Whole Message = yes
Log Spam = yes
SpamAssassin User State Dir =/var/spool/MailScanner/spamassassin
```

Clase aspam::clamd

/etc/puppet/modules/aspam/manifests/clamd.pp

```
# clam.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs and configure clamd
#

class aspam::clamd{

	package {
		'clamd':
			ensure =>	present,
			require	=>
	Class["tunning::cron","tunning::file"],
		}

	service {
		'clamd':
			enable		=>	true,
			ensure		=>	running,
			require		=>	Package['clamd'],
	}

	cron	{'freshclam':
		command		=>	'/usr/bin/freshclam',
		user	=>	root,
		hour	=>	1,
		minute	=>	0,
	}

	user	{'clam':
		ensure	=>	present,
		groups	=>	['postfix'],
		notify	=>	Service['clamd'],
	}

}
```

/root/clamd.avc

```
type=AVC msg=audit(1324694054.476:107): avc: denied { read } for
pid=12204
comm="clamd" name="12154" dev=dm-4 ino=7830
scontext=unconfined_u:system_r:clamd_t:s0
tcontext=unconfined_u:object_r:var_spool_t:s0
tclass=dir
type=AVC msg=audit(1324703243.566:233): avc: denied { getattr } for
pid=14582
comm="clamd" path="/var/spool/MailScanner/incoming/14516/1/neicar.com"
dev=dm-4 ino=2464 scontext=unconfined_u:system_r:clamd_t:s0
tcontext=unconfined_u:object_r:var_spool_t:s0 tclass=file
type=AVC msg=audit(1324703626.383:239): avc: denied { read } for
pid=14931
comm="clamd" name="neicar.com" dev=dm-4 ino=2466
```

```
scontext=unconfined_u:system_r:clamd_t:s0
tcontext=unconfined_u:object_r:var_spool_t:s0
tclass=file
type=AVC msg=audit(1324781266.918:393): avc: denied { open } for
pid=22670
comm="clamd" name="neicar.com" dev=dm-4 ino=2814
scontext=unconfined_u:system_r:clamd_t:s0
tcontext=unconfined_u:object_r:var_spool_t:s0
tclass=file
```

/root/clamd.te

```
module clamd 1.0;
require {
type var_spool_t;
type clamd_t;
class dir read;
class file { read getattr open }; }
#============= clamd_t ==============
allow clamd_t var_spool_t:dir read;
allow clamd_t var_spool_t:file open;
allow clamd_t var_spool_t:file { read getattr };
```

Clase aspam::spamassassin

/etc/puppet/modules/aspam/manifests/spamassassin.pp

```
# spanassassin.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs and configure clamd
#

class aspam::spamassassin {

	package {
		'spamassassin':
			ensure => present,
	}

	cron   {
		'sa-update':
			command      =>     '/usr/bin/sa-update',
			user   =>     root,
			hour   =>     1,
			minute =>     0,

	}
}
```

Plantilla de servidor VPN Roadwarrior

/etc/puppet/modules/openvpn/manifests/init.pp

```
# openvpn.pp
# Copyright (C) 2011 - Sergio Pena <kekio.one@gmail.com>
# See LICENSE for the full license granted to you.

# Installs & configure nrpe
#
class openvpn {

       package {
              'openvpn':
                     ensure =>     'installed',
       }
       service {
              'openvpn':
                     ensure =>     running,
                     enable =>     true,
       }

       file {
              'openvpn.cfg':
                     ensure =>     file,
                     path   =>     '/etc/openvpn/openvpn.cfg',
                     content      =>     template('openvpn.cfg');
              'ca.crt':
                     ensure =>     file,
                     path   =>     '/etc/openvpn/ca.crt',
                     source =>     ["puppet:///modules/openvpn/
$::fqdn/keys/ca.crt"];
              'server.crt':
                     ensure =>     file,
                     path   =>     '/etc/openvpn/server.crt',
                     source =>     ["puppet:///modules/openvpn/
$::fqdn/keys/server.crt"];
              'server.key':
                     ensure =>     file,
                     path   =>     '/etc/openvpn/server.key',
                     source =>     ["puppet:///modules/openvpn/
$::fqdn/keys/server.key"];
              'dh1024.pem':
                     ensure =>     file,
                     path   =>     '/etc/openvpn/dh1024.pem',
                     source =>     ["puppet:///modules/openvpn/
$::fqdn/keys/dh1024.pem"];
       }

}
```

/etc/puppet/modules/openvpn/templates/openvpn.cfg

```
port 1194
proto tcp
dev tun
ca ca.crt
cert server.crt
key server.key
dh dh1024.pem
server <%= server %> 255.255.255.0
ifconfig-pool-persist ipp.txt
keepalive 10 120
```

```
        comp-lzo
        persist-key
        persist-tun
        status server_status.log
verb 3
```