Universidad Autónoma Metropolitana
Unidad Azcapotzalco
División de Ciencias Básicas e Ingeniería
Licenciatura en Ingeniería en Computación

Sistema de monitoreo y respuesta a incidentes en activos de red corporativos.

Modalidad: Estancia Profesional

Trimestre 2022 Periodo Invierno

María Fernanda Bravo Correa
Matricula: 2173037862

Titular
M. en C. José Alfredo Estrada Soto

Co - asesor
M.S.I. Eduardo Mendoza Valdez

Mayo de 2022

Yo, José Alfredo Estrada Soto, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

_____

M. en C. José Alfredo Estrada Soto

Yo, Eduardo Mendoza Valdez, declaro que aprobé el contenido del presente Reporte de Proyecto de Integración y doy mi autorización para su publicación en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

M.S.I. Eduardo Mendoza Valdez

Yo, María Fernanda Bravo Correa, doy mi autorización a la Coordinación de Servicios de Información de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, para publicar el presente documento en la Biblioteca Digital, así como en el Repositorio Institucional de UAM Azcapotzalco.

María Fernanda Bravo Correa

Resumen

El informe del proyecto “Sistema de monitoreo y respuesta a incidentes en activos de red corporativos” contiene el análisis, desarrollo y solución hacia la necesidad de la empresa T&B Talent de implementar una herramienta que permita monitorear activos de forma remota y en tiempo real, ya que el monitoreo es un componente esencial para las actividades desarrolladas dentro de la empresa.

Para ello, dentro de un servidor de producción, se implementó el sistema Nagios, que permite habilitar y deshabilitar características o componentes, de esta forma se adapta a los requerimientos de los usuarios.

Se realizó la instalación de PNP, una herramienta que no solo genera gráficos con los datos recopilados de los dispositivos, sino que optimiza el proceso de generación de reportes en PDF.

Además se desarrollaron las configuraciones necesarias para el envío de notificaciones en caso de que el estado de algún dispositivo o servicio monitoreado, presente un estado fuera de los parámetros determinados como normales.

Por último, se realizó la configuración de dos equipos de prueba para corroborar el correcto monitoreo, de las tecnologías principales, posteriormente se configurarán los activos de los clientes para finalizar la implementación del sistema.

Contenido

Índice de ilustraciones.

Introducción

Las redes de datos según su naturaleza, pueden ser redes cableadas o inalámbricas, dentro de su arquitectura, pueden contener elementos de red como enrutadores de tráfico, switches, firewalls, servidores entre otros, estos elementos deben funcionar de forma correcta para garantizar que los servicios de red sean eficientes y seguros.

Debido al constante crecimiento de las redes de datos, estas se han vuelto mucho más complejas y heterogéneas, haciendo que su control y seguimiento se tornen complicados, esto se debe a que actualmente, las redes no sólo se usan para enviar correos electrónicos o para transferir archivos, sino que se usan para dar soporte a aplicaciones robustas y servicios de vital importancia para el correcto funcionamiento de las empresas [7].

Para tratar de garantizar el buen funcionamiento de los servicios de red, se realizan monitoreos a las redes de datos, para ello, se han desarrollado sistemas de monitoreo de red, que incluyen herramientas de software y hardware para hacer un seguimiento de diversos aspectos de la red y su funcionamiento, tales como el tráfico, ancho de banda y tiempo de actividad. Estos sistemas pueden detectar dispositivos y otros elementos que interactúen con la red, además, proporcionan actualizaciones del estado de esta y sus componentes [1].

Los administradores de red hacen uso de los sistemas de monitoreo de red para detectar rápidamente las fallas de dispositivos o conexiones, problemas como los cuellos de botella de tráfico que limitan el flujo de datos. Estos sistemas, pueden alertar a los administradores sobre los problemas que se puedan presentar, además, proporcionan informes de rendimiento de los componentes y la red. Con el análisis de estos informes, los administradores pueden anticiparse a las necesidades de la organización, respecto a la actualización o implementación, de la infraestructura de red [1].

Antecedentes

Antecedentes de proyectos de la UAM Azcapotzalco

“Monitoreo del funcionamiento de motores de C.A. empleados en procesos industriales mediante un sistema en red centralizado a una computadora personal.” [2].

Para la realización de este proyecto se implementó una red de sensores de forma distribuida para poder monitorear de forma centralizada el estado de los motores usados dentro de una planta industrial. Con los valores obtenidos se logró hacer un análisis del comportamiento de los motores y con ello se optimizaron ciertos procesos.

“Sistema de Monitoreo de Alarmas de Emergencia.” [3].

Para el desarrollo de este proyecto se realizó el monitoreo de cámaras, botones de pánico y equipos de cómputo de la central de emergencias con el propósito de atender de forma oportuna a las peticiones por parte de los usuarios. Para el monitoreo de los botones de pánico se hizo uso de un PLC para minimizar la posibilidad de errores.

“Sistema de monitoreo y registro de temperaturas para incubadoras.” [4].

En este proyecto se realizó un sistema que registra en tiempo real la temperatura de las incubadoras de los hospitales, y si se detecta una anomalía en los valores obtenidos, se activa una alarma para que se atienda y corrija la anomalía, con lo que se garantiza un estado apropiado para que los bebés.

Antecedentes de proyectos externos

“Servidor de control de dispositivos y servicios mediante el protocolo SNMP para la red de datos en CELEC.E.P. unidad de negocio Hidroagoyán.” [5]

En este proyecto se implementó un servidor de control de dispositivos y servicios de datos, haciendo uso del protocolo SNMP, para monitorear el estado y funcionamiento de la red de Hidro-Agoyán.

“Implementación de monitoreo de red utilizando los Protocolos icmp y snmp.” [6].

Para este proyecto, se implementó un servidor para monitorear la red de la Universidad Estatal Península de Santa Elena mediante protocolo ICMP y SNMP para supervisar el estado y rendimiento de la red de datos.

“DSpace ESPOCH.: Análisis del protocolo SNMPv3 para el desarrollo de un prototipo de monitoreo de red segura.” [7].

El proyecto realiza una investigación sobre el protocolo SNMPv3 y sus características, evalúa si dichas características cumplen con los requerimientos de seguridad que sus versiones anteriores no lograron cumplir. Además, hace una simulación de cómo se comportaría dicho protocolo en una red de datos.

Justificación

El monitoreo de redes de datos es de gran importancia, notificando y ayudando a prevenir fallas en el funcionamiento de la red, para su posterior atención.

Al reunir los monitoreos de las redes, dentro de un solo panel de información, se podrán visualizar las notificaciones y alertas de los equipos pertenecientes de las mismas en tiempo real, de esta forma, se busca optimizar el tiempo y esfuerzo requeridos para consultar y atender dichas alertas, haciendo más eficientes los monitoreos.

Objetivos

Objetivo General

Obtener información en tiempo real de los activos de red a monitorear, para responder y atender los incidentes de una manera más eficiente.

Objetivos Específicos

- Desarrollar una etapa de identificación de los activos a monitorear.
- Desarrollar una etapa de recopilación de los datos de los activos ya identificados.
- Desarrollar una etapa para implementar una base de datos que almacene los datos de los activos.
- Desarrollar una etapa de acoplamiento del sistema de monitoreo de redes para visualizar la información recabada.

Marco Teórico

Para la realización de este proyecto, fue necesario familiarizarse con conceptos como:

- SNMP: El protocolo simple de administración de redes, es un protocolo de capa de aplicación, usado para intercambiar información de administración entre dispositivos de red, además forma parte de los protocolos TCP/IP [8].
- Agente SNMP: Es un programa que permite recopilar la información de los equipos administrados y lo pone a disposición del administrador SNMP [8].
- Administrador SNMP: Es la entidad encargada de comunicarse con los dispositivos de red implementados por el agente SNMP. Es un equipo que ejecuta sistemas de administración de red [8].
- MIB: Las bases de datos de información de administración, son aquellas en las que los agentes SNMP vacían la información de los dispositivos administrados. Contienen un conjunto de valores estadísticos y de control definidos para nodos de hardware de una red [8].
- OID: Los identificadores de objeto u OID son los elementos que componen a las MIB. Cada identificador es único, y describe características específicas de un dispositivo administrado [8].
- Monitoreo de red: Es una herramienta que proporciona información a los administradores de red para determinar si la red funciona de manera óptima o no [9].
- Trap: Son mensajes de agentes SNMP enviados cuando sucede un imprevisto dentro de un dispositivo administrado [8].

Además de capacitarse en herramientas de software tales como:

- Sistema de monitoreo de redes: Son herramientas que permiten recopilar y analizar el estado de las redes de datos y sus componentes. Los datos recopilados se presentan a los administradores de la red, para que ellos interpreten los datos y tomen decisiones a favor de la red y sus dispositivos.
- Zabbix: Es un sistema de monitoreo de redes que permite monitorear la capacidad, rendimiento, y disponibilidad de equipos, aplicaciones y bases de datos [10].
- Nagios: Es un sistema de monitorización de redes, es de código abierto, supervisa equipos y servicios, alertando cuando su comportamiento no es el

deseado. Permite la monitorización de servicios de red como SMTP, POP3, HTTP, SNMP, además de la monitorización de los recursos de sistemas hardware como carga del procesador, uso de los discos, memoria, estado de los puertos además permite programar plugins específicos para nuevos sistemas [11].

Desarrollo del proyecto

Requisitos previos.

Para la implementación del sistema de monitoreo, se hizo uso una máquina virtual con las siguientes características:

- 150 GB de disco duro
- 8 GB de RAM
- 4 Cores
- SO Rocky Linux

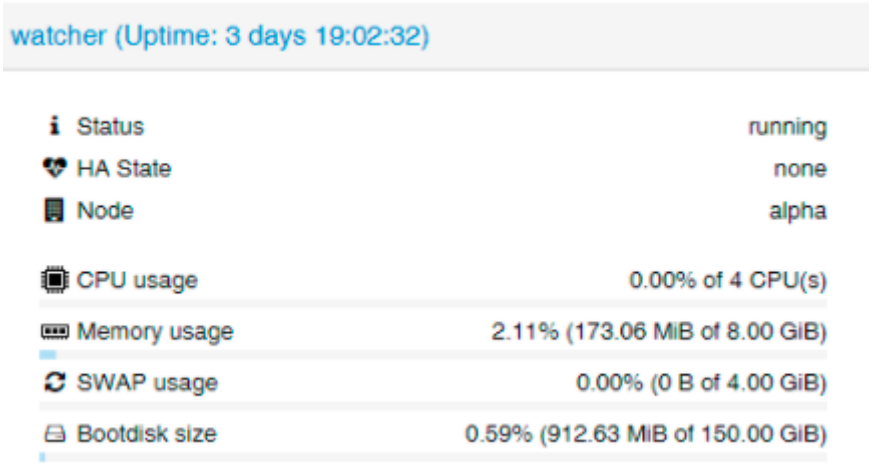

Ilustración 1. Características de la VM.

Para poder acceder a la máquina virtual de forma remota fue necesario habilitar el protocolo SSH, los comandos necesarios para la instalación y configuración son:

```
sudo yum -y install openssh-server openssh-clients
```

Iniciar el protocolo de comunicación SSH:

```
sudo systemctl start sshd
```

Verificar el estado del servicio:

```
sudo systemctl status sshd
```

Nagios.

Nagios es un sistema de monitoreo de redes que es ampliamente usado y de código abierto. Permite monitorear tanto hardware y software, por lo que se eligió como el sistema de monitoreo para esta instalación.

Para instalar Nagios Core v.4.4.6, se siguió la documentación oficial de Nagios, a continuación se describen los pasos necesarios para dicha instalación.

1. Acceder como usuario root.

```
su
```

2. Instalar los paquetes de requisitos previos para la instalación.

```
dnf install -y gcc glibc glibc-common perl httpd php wget gd gd-devel
dnf install openssl-devel
dnf update -y
```

3. Descarga de los recursos de Nagios Core.

```
cd /tmp
wget -O nagioscore.tar.gz
https://github.com/NagiosEnterprises/nagioscore/archive/nagios-
4.4.6.tar.gz
tar xzf nagioscore.tar.gz
```

4. Compilar.

```
cd /tmp/nagioscore-nagios-4.4.6/
./configure
make all
```

5. Creación de usuario y grupo.

```
make install-groups-users
usermod -a -G nagios apache
```

6. Instalación de los archivos binarios, los CGI’s y los archivos HTML.

```
make install
```

7. Instalación y configuración del servidor Daemon.

```
make install-daemoninit
systemctl enable httpd.service
```

8. Instalación del modo comando.

```
make install-commandmode
```

9. Instalación de los archivos de configuración.

```
make install-config
```

10. Instalación de los archivos de la configuración de Apache:

```
make install-webconf
```

11. Configuración del Firewall para que habilitar el tráfico de entrada por el puerto 80:

```
firewall-cmd --zone=public --add-port=80/tcp
firewall-cmd --zone=public --add-port=80/tcp –permanent
```

12. Creación del usuario de Apache:

```
htpasswd -c /usr/local/nagios/etc/htpasswd.users nagiosadmin
```

13. Inicializar el servidor web de Apache:

```
systemctl start httpd.service
```

14. Inicializar el servicio Daemon.

```
systemctl start nagios.service
```

Plugins de Nagios.

Nagios Core hace uso de complementos para que funcione de forma correcta. Los pasos que se presentan a continuación fueron los necesarios para instalar los plugins básicos para Nagios Core.

1. Instalación de los paquetes previos.

```
yum install -y gcc glibc glibc-common make gettext automake autoconf
wget openssl-devel net-snmp net-snmp-utils epel-release
yum --enablerepo=powertools,epel install perl-Net-SNMP
```

2. Descargar los recursos.

```
cd /tmp
wget --no-check-certificate -O nagios-plugins.tar.gz
https://github.com/nagios-plugins/nagios-plugins/archive/release-
2.3.3.tar.gz
tar zxf nagios-plugins.tar.gz
```

3. Instalación de los recursos.

```
cd /tmp/nagios-plugins-release-2.3.3/
./tools/setup
./configure
make
make install
```

Para corroborar el estado de la instalación, fue necesario acceder desde un navegador web a la ruta:

http://IP-del-servidor/nagios

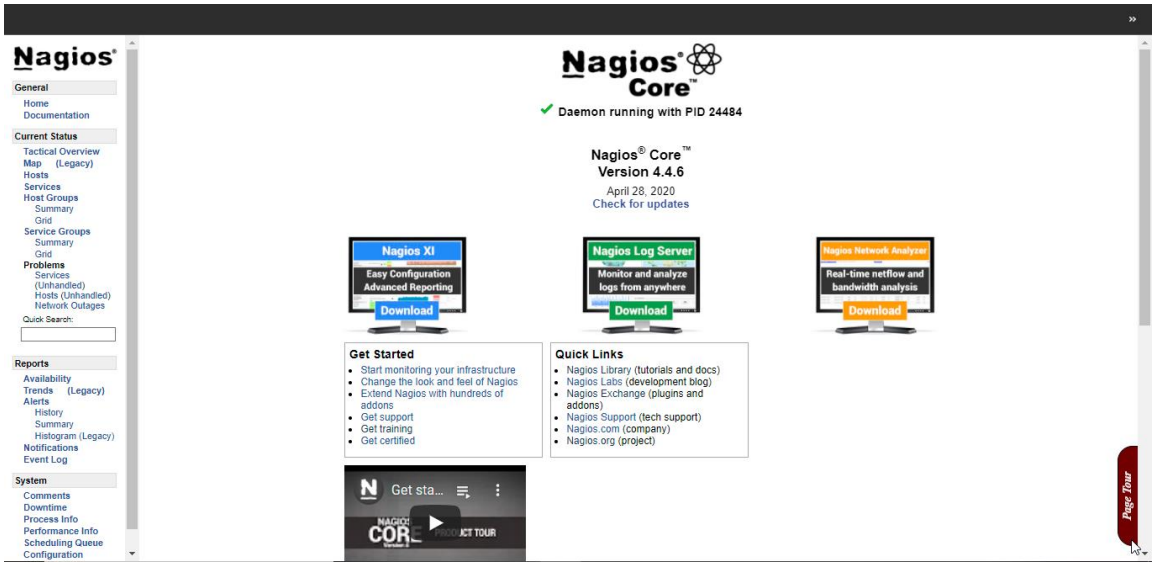

Ilustración 2. Interfaz web de Nagios Core.

PNP4Nagios.

Aunque Nagios realiza el monitoreo de los equipos, en ocasiones es necesario contar con una herramienta que permita obtener gráficas y reportes de los servicios monitoreados.

PNP4Nagios es esa herramienta, ya que es capaz de analizar los datos de los monitoreos realizados por Nagios y a partir de ellos, genera gráficas de dichos servicios.

A continuación se describen los pasos necesarios para la instalación y configuración de PNP4Nagios.

1. Instalación de los paquetes necesarios.

```
sudo apt-get update
sudo apt-get install -y rrdtool librrdp-perl php-gd php-xml
cd /tmp
wget -O pnp4nagios.tar.gz
https://github.com/lingej/pnp4nagios/archive/0.6.26.tar.gz
tar xzf pnp4nagios.tar.gz
cd pnp4nagios-0.6.26/
./configure --with-httpd-conf=/etc/apache2/sites-enabled
make all
make fullinstall
make install-webconf
make install-config
make install-init
systemctl daemon-reload
systemctl restart apache2
```

2. Acceso desde el navegador web a la ruta:

http://IP-del-servidor/pnp4nagios.

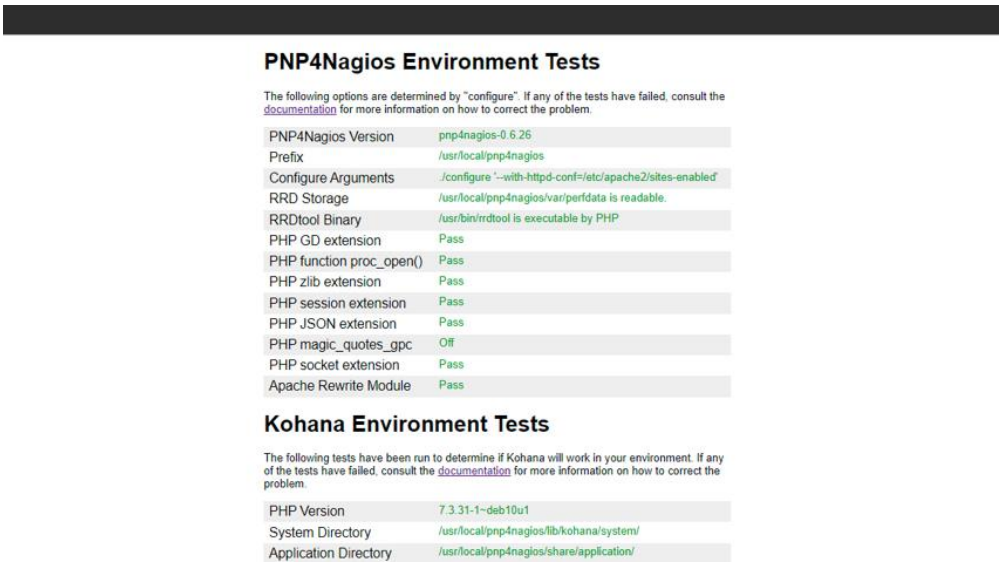

Ilustración 3. Interfaz de PNP4Nagios.

3. La pantalla anterior es una forma de comprobar que la instalación fue correcta, por lo que se borró el archivo de instalación install.php.

```
rm -f /usr/local/pnp4nagios/share/install.php
```

4. Para la configuración de PNP, fue necesario editar tres archivos, el primero es Input.php ubicado en la ruta: `/usr/local/pnp4nagios/lib*/kohana/system/libraries/`, dentro del archivo, se ubicó el siguiente fragmento de código:

```
// magic_quotes_runtime is enabled
if (get_magic_quotes_runtime())
{
      set_magic_quotes_runtime(0);
      ......................
      // magic_quotes_gpc is enabled
      if (get_magic_quotes_gpc())
      ....................
      ..................... http://php.net/magic_quotes');
}
```

Y anidarlo dentro de otra sentencia if, de la siguiente forma:

```
// magic_quotes_runtime is enabled
if (version_compare(PHP_VERSION, '5.3.0', '<')) {
      if (get_magic_quotes_runtime())
      {
          set_magic_quotes_runtime(0);
          ....................
          // magic_quotes_gpc is enabled
          if (get_magic_quotes_gpc())
          ..................
          ................... http://php.net/magic_quotes');
      }
}
```

5. Se editó el archivo data.php ubicado en la ruta:

   usr/local/pnp4nagios/share/application/models/, se sustituyó la línea:

```
if(sizeof($pages)>0){
```

   por:

```
if(is_array($pages) && sizeof($pages) > 0 ){
```

6. Se editó el archivo json.php ubicado en la ruta

   /usr/local/pnp4nagios/share/application/lib/, se sustituyó la línea:

```
cass Services_JSON
```

   por:

```
class __Services_JSON
```

7. Fue necesario integrar PNP con Nagios, para ello, se respaldó el archivo nagios.cfg, y se habilitaron los complementos que devuelven los datos del rendimiento a aplicaciones externas y se configuró PNP para que haga uso de los datos:

   7.1. Respaldo el archivo *"nagios.cfg"* como *"nagios.cfg.ori"*.

```
cp /usr/local/nagios/etc/nagios.cfg /usr/local/nagios/etc/nagios.cfg.ori
```

7.2. Sustitución de *"process_performance_data=0"* por *"process_performance_data=1"*.

```
sed -i 's/process_performance_data=0/process_performance_data=1/g'
/usr/local/nagios/etc/nagios.cfg
```

7.3. Sustitución de *"#host_perfdata_file="* por *"host_perfdata_file="*.

```
sed -i 's/#host_perfdata_file=/host_perfdata_file=/g'
/usr/local/nagios/etc/nagios.cfg
```

7.4. Sustitución de *"host_perfdata_file="* por *"host_perfdata_file= /usr/local/pnp4nagios/var/host-perfdata"*.

```
sed -i
's/^host_perfdata_file=.*/host_perfdata_file=\/usr\/local\/pnp4nagios\/v
ar\/host-perfdata/g' /usr/local/nagios/etc/nagios.cfg
```

7.5. Sustitución de *"#host_perfdata_file_template="* por *"host_perfdata_file_template=DATATYPE:HOSTPERFDATA\TIMET::$TIMET$\HOSTNAME::$HOSTNAME$\HOSTPERFDATA::$HOSTPERFDATA$\HOSTCHECKCOMMAND::$HOSTCHECKCOMMAND$\HOSTSTATE::$HOSTSTATE$\HOSTSTATETYPE::$HOSTSTATETYPE$"*.

```
sed -i
's/^#host_perfdata_file_template=.*/host_perfdata_file_template=DATATYPE
:HOSTPERFDATA\\tTIMET::$TIMET$\\tHOSTNAME::$HOSTNAME$\\tHOSTPERFDATA::$H
OSTPERFDATA$\\tHOSTCHECKCOMMAND::$HOSTCHECKCOMMAND$\\tHOSTSTATE::$HOSTST
ATE$\\tHOSTSTATETYPE::$HOSTSTATETYPE$/g'
/usr/local/nagios/etc/nagios.cfg
```

7.6. Sustitución de *"#host_perfdata_file_mode="* por *"host_perfdata_file_mode="*.

```
sed -i 's/#host_perfdata_file_mode=/host_perfdata_file_mode=/g'
/usr/local/nagios/etc/nagios.cfg
```

7.7. Sustitución de *"#host_perfdata_file_processing_interval="* por *"host_perfdata_file_processing_interval=15"*.

```
sed -i
's/^#host_perfdata_file_processing_interval=.*/host_perfdata_file_proces
sing_interval=15/g' /usr/local/nagios/etc/nagios.cfg
```

7.8. Sustitución de *"#host_perfdata_file_processing_command="* por *"host_perfdata_file_processing_command=process-host-perfdata-file-bulk-npcd"*.

```
sed -i
's/^#host_perfdata_file_processing_command=.*/host_perfdata_file_process
ing_command=process-host-perfdata-file-bulk-npcd/g'
/usr/local/nagios/etc/nagios.cfg
```

7.9. Sustitución de *"#service_perfdata_file="* por *"service_perfdata_file="*.

```
sed -i 's/#service_perfdata_file=/service_perfdata_file=/g'
/usr/local/nagios/etc/nagios.cfg
```

7.10. Sustitución de *"service_perfdata_file="* por *"service_perfdata_file=/usr/local/pnp4nagios/var/service-perfdata"*.

```
sed -i
's/^service_perfdata_file=.*/service_perfdata_file=\/usr\/local\/pnp4nag
ios\/var\/service-perfdata/g' /usr/local/nagios/etc/nagios.cfg
```

7.11. Sustitución de *"#service_perfdata_file_template="* por *"service_perfdata_file_template=DATATYPE::SERVICEPERFDATA\TIMET::$TIMET$\HOSTNAME::$HOSTNAME$\SERVICEDESC::$SERVICEDESC$\SERVICEPERFDATA::$SERVICEPERFDATA$\SERVICECHECKCOMMAND::$SERVICECHECKCOMMAND$\HOSTSTATE::$HOSTSTATE$\HOSTSTATETYPE::$HOSTSTATETYPE$\SERVICESTATE::$SERVICESTATE$\SERVICESTATETYPE::$SERVICESTATETYPE$"*.

```
sed -i
's/^#service_perfdata_file_template=.*/service_perfdata_file_template=DA
TATYPE::SERVICEPERFDATA\\tTIMET::$TIMET$\\tHOSTNAME::$HOSTNAME$\\tSERVIC
EDESC::$SERVICEDESC$\\tSERVICEPERFDATA::$SERVICEPERFDATA$\\tSERVICECHECK
COMMAND::$SERVICECHECKCOMMAND$\\tHOSTSTATE::$HOSTSTATE$\\tHOSTSTATETYPE:
:$HOSTSTATETYPE$\\tSERVICESTATE::$SERVICESTATE$\\tSERVICESTATETYPE::$SER
VICESTATETYPE$/g' /usr/local/nagios/etc/nagios.cfg
```

7.12. Sustitución de *"#service_perfdata_file_mode="* por *"service_perfdata_file_mode="*.

```
sed -i 's/#service_perfdata_file_mode=/service_perfdata_file_mode=/g'
/usr/local/nagios/etc/nagios.cfg
```

7.13. Sustitución de *"#service_perfdata_file_processing_interval="* por *"service_perfdata_file_processing_interval=15"*.

```
sed -i
's/^#service_perfdata_file_processing_interval=.*/service_perfdata_file_
processing_interval=15/g' /usr/local/nagios/etc/nagios.cfg
```

7.14. Sustitución de *"#service_perfdata_file_processing_command="* por *"service_perfdata_file_processing_command=process-service-perfdata-file-bulk-npcd"*.

```
sed -i
's/^#service_perfdata_file_processing_command=.*/service_perfdata_file_p
rocessing_command=process-service-perfdata-file-bulk-npcd/g'
/usr/local/nagios/etc/nagios.cfg
```

8. Se agregaron los siguientes comandos dentro del archivo *commands.cfg* ubicado en la ruta */usr/local/nagios/etc/objects/* con esto, se indicó que los datos de los monitoreos sean procesados por la aplicación de PNP

```
define command {
    command_name    process-service-perfdata-file-bulk-npcd
    command_line    /bin/mv /usr/local/pnp4nagios/var/service-
perfdata/usr/local/pnp4nagios/var/spool/service-perfdata.$TIMET$
}
```

```
define command {
    command_name    process-host-perfdata-file-bulk-npcd
    command_line    /bin/mv /usr/local/pnp4nagios/var/host-perfdata
/usr/local/pnp4nagios/var/spool/host-perfdata.$TIMET$
}
```

9. Para ver reflejados los cambios realizados, fue necesario reiniciar los servicios con:

```
systemctl enable npcd.service
systemctl start npcd.service
systemctl restart httpd.service
/usr/local/pnp4nagios/bin/npcd -d -f /usr/local/pnp4nagios/etc/npcd.cfg
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
systemctl restart nagios.service
```

10. El archivo templates.cfg ubicado en la ruta *usr/local/nagios/etc/objects/* contiene las plantillas genéricas de host, servicios, contactos entre otras, se agregaron las siguientes plantillas: una es para hosts que usen PNP y la otra es para los servicios que usen PNP.

```
define host {
   name       host-pnp
   action_url /pnp4nagios/index.php/graph?host=$HOSTNAME$&srv=_HOST_
   register   0
}
```

```
define service {
   name       service-pnp
   action_url
/pnp4nagios/index.php/graph?host=$HOSTNAME$&srv=$SERVICEDESC$
   register   0
}
```

11. Dentro del mismo archivo, se modificó el campo *"use"* en las declaraciones de *generic-host* y *generic-service*, con ello se indica que para cualquier host o servicio que haga uso de *generic-host* o *generic-service,* sus datos serán recopilados por PNP.

    Para el caso de *generic-host* se declaró la línea:

```
use        host-pnp
```

```
define host {
    name                            generic-nost
    use                             host-pnp
    notifications_enabled           1
    event_handler_enabled           1
    flap_detection_enabled          1
    process_perf_data               1
    retain_status_information       1
    retain_nonstatus_information    1
    notification_period             24x7
    register                        0
}
```

Ilustración 4. Definición de generic-host modificada.

Para el caso de *generic-service*, se declaró la línea:

```
use        service-pnp
```

```
define service {
    name                            generic-service
    use                             service-pnp
    active_checks_enabled           1
    passive_checks_enabled          1
    parallelize_check               1
    obsess_over_service             1
    check_freshness                 0
    notifications_enabled           1
    event_handler_enabled           1
    flap_detection_enabled          1
    process_perf_data               1
    retain_status_information       1
    retain_nonstatus_information    1
    is_volatile                     0
    check_period                    24x7
    max_check_attempts              3
    check_interval                  10
    retry_interval                  2
    contact_groups                  admins
    notification_options            w,u,c,r
    notification_interval           60
    notification_period             24x7
    register                        0
}
```

Ilustración 5 Definición de generic-service modificada.

12. Se reinició Nagios para corroborar la integrado con PNP.

```
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
systemctl restart nagios
```

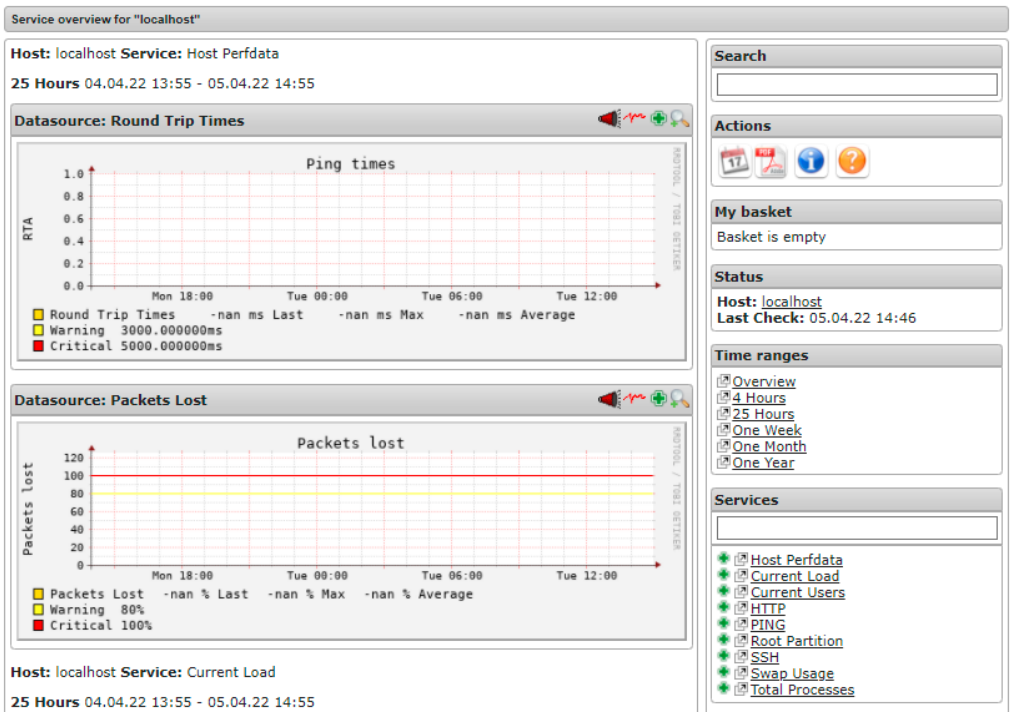

Ilustración 6. Integración de Nagios y PNP4Nagios.

Reportes en PDF.

Fue necesario automatizar el proceso para generar reportes del estado de los equipos, a continuación se explican los pasos necesarios para generar dos tipos de reportes en formato PDF con plantillas distintas, todo esto se realizó a través de la interfaz web de PNP4Nagios.

1. Para generar dos tipos de reportes con formato PDF, se modificó el archivo *icon_box.php* ubicado en `/usr/local/pnp4nagios/share/application/views/`, se duplicó el siguiente fragmento de código:

```
if($this->config->conf['use_fpdf'] == 1 && ( $position == "graph" || $position
== "special") ){
    echo "<a title=\"".Kohana::lang('common.title-pdf-link')."\"
href=\"".url::base(TRUE)."pdf".$qsa."\"><img class=\"icon\"
src=\"".url::base()."media/images/pdf.png\"></a>\n";
}

if($this->config->conf['use_fpdf'] == 1 && $position == "basket"){
    echo "<a title=\"".Kohana::lang('common.title-pdf-link')."\"
href=\"".url::base(TRUE)."pdf/basket/".$qsa."\"><img class=\"icon\"
src=\"".url::base()."media/images/pdf.png\"></a>\n";
}

if($this->config->conf['use_fpdf'] == 1 && $position == "page"){
    echo "<a title=\"".Kohana::lang('common.title-pdf-link')."\"
href=\"".url::base(TRUE)."pdf/page/".$this->page.$qsa."\"><img class=\"icon\"
src=\"".url::base()."media/images/pdf.png\"></a>\n";
}
```

2. En el fragmento duplicado, se reemplazó *"pdf"* por *"pdf1"* como se muestra a continuación.

```
if($this->config->conf['use_fpdf'] == 1 && ( $position == "graph" || $position
== "special") ){
    echo "<a title=\"".Kohana::lang('common.title-pdf-link')."\"
href=\"".url::base(TRUE)."pdf1".$qsa."\"><img class=\"icon\"
src=\"".url::base()."media/images/pdf.png\"></a>\n";
}

if($this->config->conf['use_fpdf'] == 1 && $position == "basket"){
    echo "<a title=\"".Kohana::lang('common.title-pdf-link')."\"
href=\"".url::base(TRUE)."pdf1/basket/".$qsa."\"><img class=\"icon\"
src=\"".url::base()."media/images/pdf.png\"></a>\n";
}

if($this->config->conf['use_fpdf'] == 1 && $position == "page"){
    echo "<a title=\"".Kohana::lang('common.title-pdf-link')."\"
href=\"".url::base(TRUE)."pdf1/page/".$this->page.$qsa."\"><img class=\"icon\"
src=\"".url::base()."media/images/pdf.png\"></a>\n";
}
```

3. En la ruta */usr/local/pnp4nagios/share/application/controllers/* se duplicó el archivo *pdf.php* y se nombró como *pdf1.php*.
4. Dentro del archivo *pdf1.php* se sustituyó *"background.pdf"* por *"background1.pdf"* y *"class Pdf_Controller"* por *"class Pdf1_Controller"*.

```
. . .  . . .
class Pdf1_Controller extends System_Controller  {
    public function __construct(){
        parent::__construct();
        $this->use_bg           = 0;
        $this->bg               = $this->config->conf['background1_pdf'];
        $this->pdf_page_size    = $this->config->conf['pdf_page_size'];
        $this->pdf_margin_left  = $this->config->conf['pdf_margin_left'];
        $this->pdf_margin_top   = $this->config->conf['pdf_margin_top'];
        $this->pdf_margin_right = $this->config->conf['pdf_margin_right'];
        // Define PDF background1 per url option
        if(isset($this->bg) && $this->bg != ""){
            if( is_readable( Kohana::config( 'core.pnp_etc_path')."/".$this->bg
) ){
                $this->bg = Kohana::config('core.pnp_etc_path')."/".$this->bg;
            }else{
                $this->bg = $this->config->conf['background1_pdf'];
            }
        }
        // Use PDF background1 if readable
        if(is_readable($this->bg)){
            $this->use_bg = 1;
        }
    }
. . . . . .
```

5. En la ruta */usr/local/pnp4nagios/etc/* se agregaron los archivos a usar como plantilla para los documentos, en este caso se identifican como *background1.pdf* y *background.pdf*

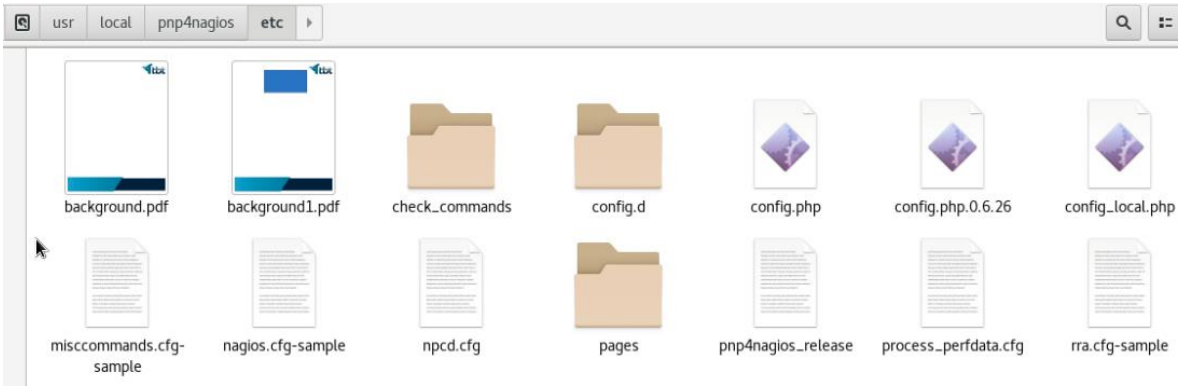

Ilustración 7. Plantillas para los diferentes reportes en PDF.

6. En la ruta */usr/local/pnp4nagios/share/application/vendor/fpdf/* se duplicó el archivo *fpdf.php*, y se nombró como *fpdf1.php*. Dentro del nuevo archivo, se sustituyó *"fpdf"* por *"fpdf1"*.

7. Se declararon 2 rutas para que cada una, dirija a la plantilla correspondiente para generar el reporte con determinado formato, para ello en el archivo *config.php* ubicado en */usr/local/pnp4nagios/etc/*, debajo de la línea 202 se agregó la línea resaltada:

```
# Use this file as PDF background.
$conf['background_pdf'] = '/usr/local/pnp4nagios/etc/background.pdf' ;
$conf['background1_pdf'] = '/usr/local/pnp4nagios/etc/background1.pdf' ;
```

8. PNP por defecto nombra los PDF como: *"pnp4nagios.pdf"*, para cambiar el nombre genérico, se modificaron los archivos *pdf.php* y *pdf1.php* ubicados en */usr/local/pnp4nagios/share/application/controllers/*, en ambos archivos, se comentó la línea 130 y se declaró la línea 131 de la siguiente forma:

```
$pdf->Output(strval($this->host).'_'.strval($this->service).'.pdf'."I");
```

Los archivos quedaron de la siguiente forma:

```
//$pdf->Output("pnp4nagios.pdf","I");
$pdf->Output(strval($this->host).'_'.strval($this->service).'.pdf',"I");
```

9. Al descargar el PDF se guarda como:

```
"host_serviciomonitoreado.pdf"
```

10. Se guardaron todos los cambios en los archivos y se reinició Nagios. El primer comando, compila todos los archivos de configuración de Nagios y en caso de existir errores, lo indica, con el segundo comando, se reinicia Nagios en caso de que no existan errores.

```
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
systemctl restart nagios
```

Una vez que se hayan realizado las configuraciones, se ve la interfaz de Nagios de la siguiente forma:

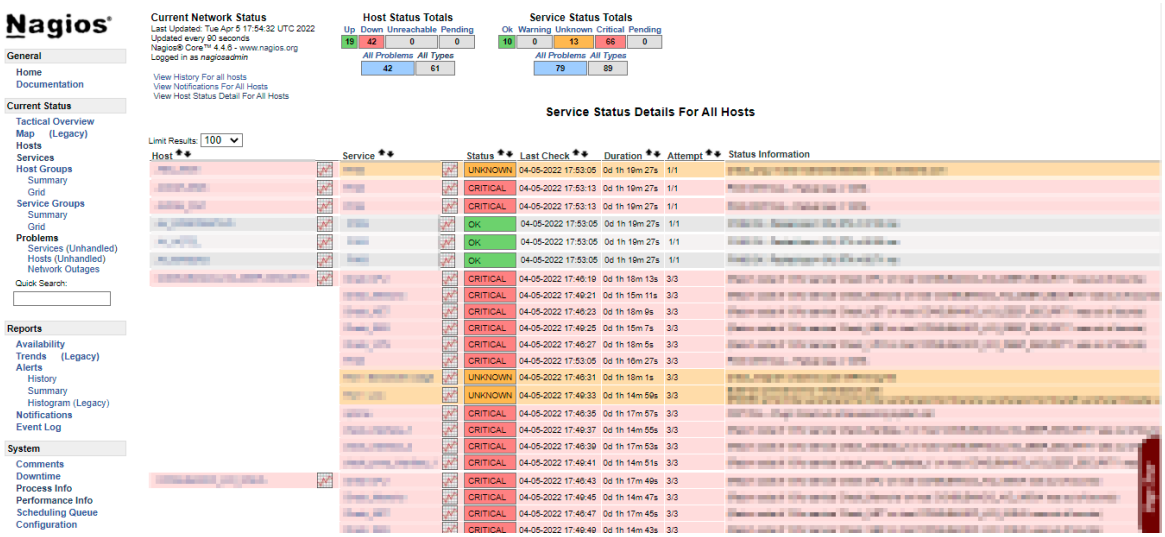

Ilustración 8. Interfaz web de Nagios.

Dentro del apartado de las gráficas de PNP, se encuentran los botones para hacer diferentes archivos PDF.

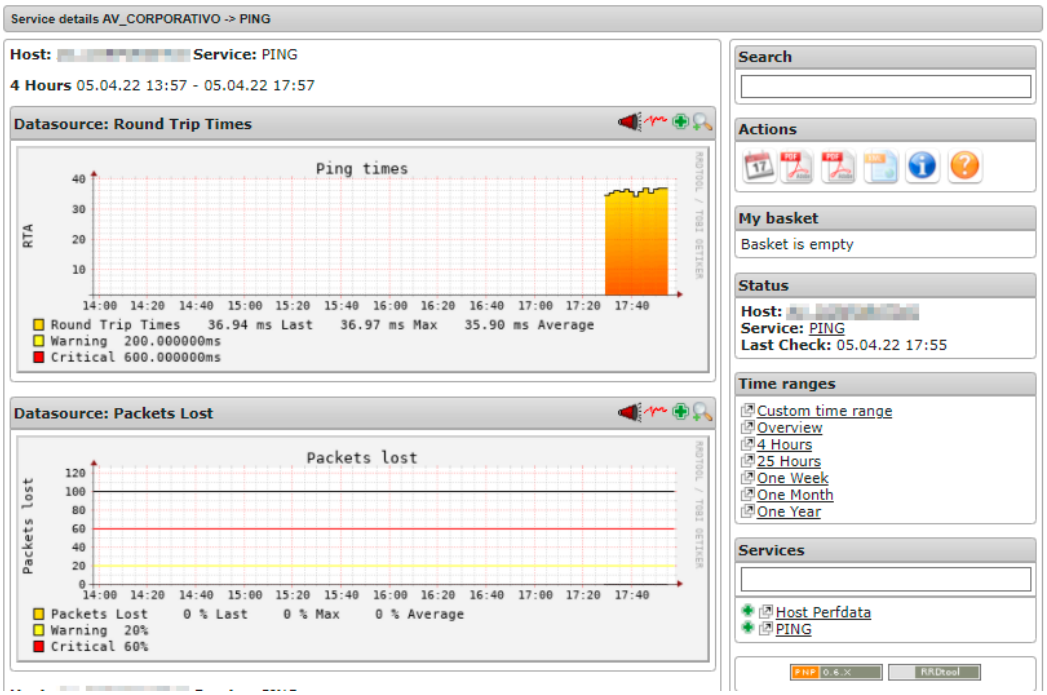

Ilustración 9. Interfaz Web de PNP4Nagios con los dos íconos para generar diferentes reportes PDF.

Configuración de host y alertas.

Plugins.

Nagios ofrece plugins para poder monitorear ciertos servicios, pero fue necesario descargar plugins adicionales para el monitoreo de ciertas tecnologías, dichos plugins se guardaron en la ruta: `/usr/local/nagios/libexec/`.

Los plugins adicionales son:

- check_nwc_health.
- check_fortigate.pl.
- check_snmp_trafiic.
- check_snmp_uptime.pl.
- check_traffic.sh.

A continuación se explica una de las formas en las que se obtuvieron los plugins, el procedimiento se aplicó para el *`plugin check_fortigate.pl`*.

1. Nagios ofrece un directorio de los plugins a través de la página:

   https://exchange.nagios.org/directory/Plugins

2. En él, se pueden buscar los complementos para diferentes tecnologías o para los distintos servicios que deseen monitorear. Para este ejemplo, se buscaron plugins para Fortinet y se desplegó una lista con diferentes plugins, se seleccionó el plugin que más se adaptó a lo requerido, en este caso, *`check_fortigate.pl`* es ese plugin.

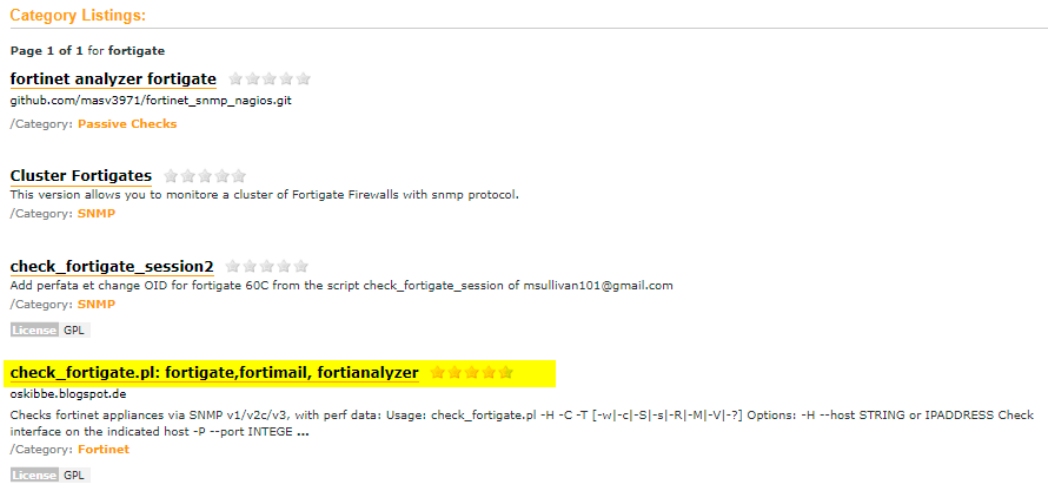

Ilustración 10. Búsqueda de plugins para Fortinet.

3. Al seleccionarlo apareció una página en la que se muestran las características del plugin, y, en este caso, aparece, un enlace hacia el repositorio externo del plugin.

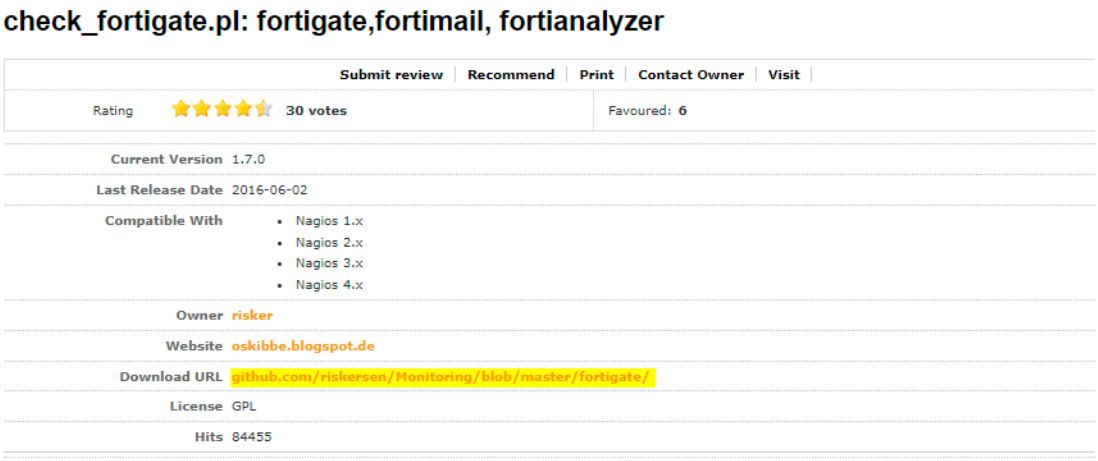

Ilustración 11. Vista del plugin check_fortigate.pl desde la página de Nagios.

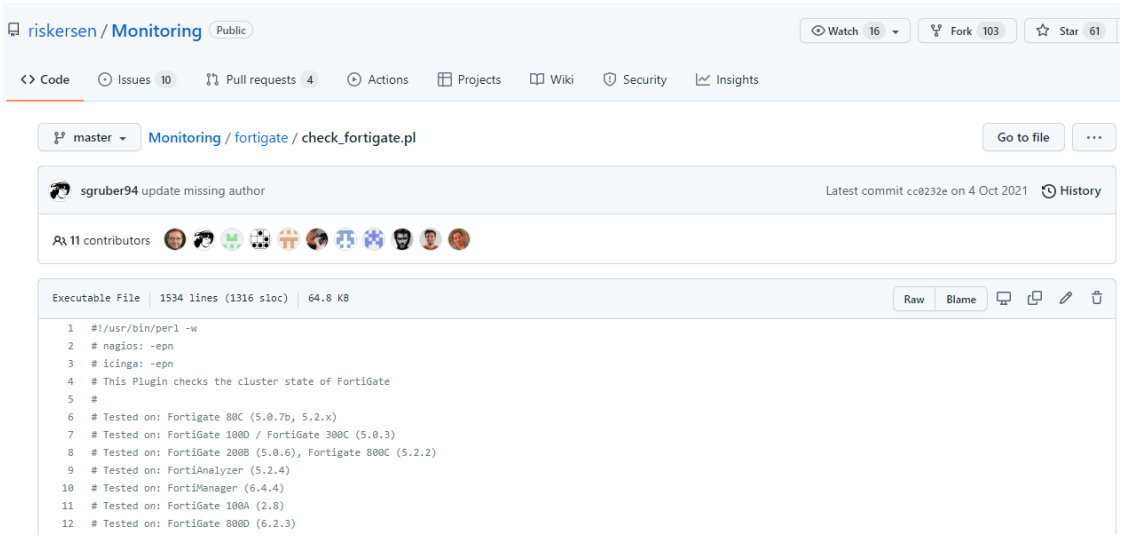

Ilustración 12. Repositorio en github del plugin check_fortigate.pl.

4. Dentro de la ruta /usr/local/nagios/libexec se creó un archivo que coincida con el nombre del plugin:

```
nano check_fortigate.pl
```

5. Fue necesario copiar el contenido del repositorio y pegarlo en el archivo que fue creado, se guardaron los cambios, y para verificar que el plugin tenga permisos de ejecución se usaron los comandos:

```
ls -la
```

```
[redacted] libexec]# ls -la
total 7984
drwxrwxr-x. 4 nagios nagios   4096 abr 26 12:27 .
drwxr-xr-x. 8 root   root       79 mar  1 12:26 ..
-rwxr-xr-x  1 root   root   215632 feb 14 22:41 check_apt
-rwxr-xr-x  1 root   root     2354 feb 14 22:41 check_breeze
-rwxr-xr-x  1 root   root   222136 feb 14 22:41 check_by_ssh
lrwxrwxrwx  1 root   root        9 feb 14 22:41 check_clamd -> check_tcp
-rwxr-xr-x  1 root   root   162560 feb 14 22:41 check_cluster
-rwxr-xr-x  1 root   root   218752 feb 14 22:41 check_dhcp
-rwxr-xr-x  1 root   root   213576 feb 14 22:41 check_dig
-rwxr-xr-x  1 root   root   375896 feb 14 22:41 check_disk
-rwxr-xr-x  1 root   root    10142 feb 14 22:41 check_disk_smb
-rwxr-xr-x  1 root   root   236864 feb 14 22:41 check_dns
-rwxr-xr-x  1 root   root   113160 feb 14 22:41 check_dummy
-rwxr-xr-x  1 root   root     5074 feb 14 22:41 check_file_age
-rwxr-xr-x  1 root   root     6512 feb 14 22:41 check_flexlm
-rw-r--r--  1 root   root    66382 abr 26 12:27 check_fortigate.pl
lrwxrwxrwx  1 root   root        9 feb 14 22:41 check_ftp -> check_tcp
```

Ilustración 13. Lista detallada de los plugins contenidos en /libexec junto con sus permisos.

6. Como se aprecia en la ilustración 13, los plugins cuentan con permisos de ejecución excepto *check_fortigate.pl*, para cambiar esto se usó:

```
chmod 755 check_fortigate.pl
```

7. Se verificó el cambio en los permisos del plugin:

```
[redacted] libexec]# ls -la
total 7984
drwxrwxr-x. 4 nagios nagios   4096 abr 26 12:27 .
drwxr-xr-x. 8 root   root       79 mar  1 12:26 ..
-rwxr-xr-x  1 root   root   215632 feb 14 22:41 check_apt
-rwxr-xr-x  1 root   root     2354 feb 14 22:41 check_breeze
-rwxr-xr-x  1 root   root   222136 feb 14 22:41 check_by_ssh
lrwxrwxrwx  1 root   root        9 feb 14 22:41 check_clamd -> check_tcp
-rwxr-xr-x  1 root   root   162560 feb 14 22:41 check_cluster
-rwxr-xr-x  1 root   root   218752 feb 14 22:41 check_dhcp
-rwxr-xr-x  1 root   root   213576 feb 14 22:41 check_dig
-rwxr-xr-x  1 root   root   375896 feb 14 22:41 check_disk
-rwxr-xr-x  1 root   root    10142 feb 14 22:41 check_disk_smb
-rwxr-xr-x  1 root   root   236864 feb 14 22:41 check_dns
-rwxr-xr-x  1 root   root   113160 feb 14 22:41 check_dummy
-rwxr-xr-x  1 root   root     5074 feb 14 22:41 check_file_age
-rwxr-xr-x  1 root   root     6512 feb 14 22:41 check_flexlm
-rwxr-xr-x  1 root   root    66382 abr 26 12:27 check_fortigate.pl
```

Ilustración 14. Lista detallada de los plugins contenidos en /libexec junto con sus permisos.

8. Fue necesario declarar los comandos necesarios dentro del archivo *commands.cfg* ubicado en */usr/local/nagios/etc/objects/*, dichos comandos deben invocar a los plugins adicionales.

   Para esta instalación, no fue necesario crear algún comando, ya que fueron declarados por el desarrollador del plugin, dentro del mismo repositorio, únicamente fue necesario pegar los comandos dentro de *commands.cfg*.

Con esto, se da por terminado el proceso de ejemplificación de la descarga de plugins adicionales.

A continuación se explica una forma de crear comandos en caso de no contar con un plugin para monitorear un equipo, este método no se usó durante esta configuración pero se explicará para complementar la información respecto a las posibles configuraciones dentro de Nagios.

9. Se pueden declarar comandos haciendo uso de OID's y del comando `check_snmp`. Para obtener los OID's de un host y guardarlos en un archivo de texto use:

```
snmpwalk -v <versiónComunidadSNMP> -c <nombreComunidad> <direccionHost>
&> nombreArchivo.txt
```

10. El archivo de texto se guardará en la ruta en la que se encuentre, y dentro de él, estará la lista con la descripción de los OID que se pueden consultar del equipo. Extraiga los que sean necesarios.
11. Para este ejemplo, se usó el OID "*ifOperStatus.2*" que indica si la interfaz 2 tiene un enlace exitoso o no.

```
IF-MIB::ifAdminStatus.1 = INTEGER: up(1)
IF-MIB::ifAdminStatus.2 = INTEGER: up(1)
IF-MIB::ifAdminStatus.3 = INTEGER: up(1)
IF-MIB::ifAdminStatus.4 = INTEGER: up(1)
IF-MIB::ifAdminStatus.5 = INTEGER: up(1)
IF-MIB::ifAdminStatus.6 = INTEGER: up(1)
IF-MIB::ifAdminStatus.7 = INTEGER: up(1)
IF-MIB::ifOperStatus.1 = INTEGER: up(1)
IF-MIB::ifOperStatus.2 = INTEGER: down(2)
IF-MIB::ifOperStatus.3 = INTEGER: down(2)
IF-MIB::ifOperStatus.4 = INTEGER: down(2)
IF-MIB::ifOperStatus.5 = INTEGER: down(2)
IF-MIB::ifOperStatus.6 = INTEGER: down(2)
IF-MIB::ifOperStatus.7 = INTEGER: down(2)
```

Ilustración 15. Lista de OID´s obtenida con el comando snmpwalk.

12. Dentro de la ruta */usr/local/nagios/etc/libexec* ejecute el siguiente comando:

```
./check_snmp -H <direccionHost> -o <OID> -C >nombreDeLaComunidad>
```

```
[root@watcher libexec]# ./check_snmp -H 10.11.11.54 -o ifOperStatus.2 -C Pruebas_SNMP_v2 10.11.11.54
SNMP OK - down(2) |
```

Ilustración 16. Implementación del comando check_snmp para consultar parámetros de un host.

13. Con el paso anterior, se verifica la sintaxis del comando y su funcionalidad.

14. Defina su comando dentro del archivo *commands.cfg* ubicado en */usr/local/nagios/etc/objects/*

```
define command {
  command_name <nombre_del_comando>
  comannd_line $USER1$/check_snmp -H $HOSTADDRESS$ -C
$_HOSTSNMP_COMMUNITY$ -o <OID>
}
```

El comando desarrollado en el punto anterior es solo un ejemplo básico, el usuario puede agregar los parámetros que considere necesarios para hacer un monitoreo a medida de lo que necesite.

Los campos *$_HOSTSNMP_COMMUNITY$* y *$HOSTADDRESS$* entre otros, son macros que Nagios ofrece para facilitar el trabajo de las configuraciones realizadas.

15. Recuerde que, antes de declarar un comando en *commands.cfg* debe asegurarse de que los plugins necesarios para la ejecución de ese comando, existan dentro del directorio de */libexec* con permisos de ejecución y posteriormente, debe invocar dicho comando en el archivo de configuración de los host.
16. Para esta instalación, fue necesario instalar librerías adicionales de Perl, para la instalación de dichas librerías, se hizo uso de la herramienta CPAN, a continuación se explica cómo se instaló dicha herramienta:

```
dnf install perl-CPAN
dnf install "@Development Tools"
```

17. Posteriormente se instalaron los complementos necesarios usando:

```
cpan json
cpan Time::HiRes
cpan File::Slurp
cpan List::Compare
```

18. Para corroborar la instalación y las configuraciones previas, se ejecutaron los siguientes comandos:

```
systemctl enable npcd.service
systemctl start npcd.service
systemctl restart httpd.service
/usr/local/pnp4nagios/bin/npcd -d -f /usr/local/pnp4nagios/etc/npcd.cfg
/usr/local/nagios/bin/nagios -v /usr/local/nagios/etc/nagios.cfg
systemctl restart nagios.service
```

Hosts.

Para esta configuración, los host a monitorear, fueron divididos por tecnología, por lo que se crearon 4 archivos de tipo .cfg, que se guardaron dentro de una carpeta nombrada switches ubicada en la ruta */usr/local/nagios/etc/*.

1. Dentro del archivo de configuración de Nagios fue necesario indicar donde están contenidos los archivos de los host. Para ello, en el archivo nagios.cfg ubicado en la ruta */usr/local/nagios/etc/* se sustituyó la línea *#cfg_dir=/usr/local/nagios/etc/switches*, por *cfg_dir=/usr/local/nagios /etc/switches*.

```
# You can also tell Nagios to process all config files (with a .cfg
# extension) in a particular directory by using the cfg_dir
# directive as shown below:

#cfg_dir=/usr/local/nagios/etc/servers
#cfg_dir=/usr/local/nagios/etc/printers
cfg_dir=/usr/local/nagios/etc/switches
#cfg_dir=/usr/local/nagios/etc/routers
```

Ilustración 17. Modificación de archivo nagios.cfg para habilitar carpeta switches.

2. Dentro de la ruta */usr/local/nagios/etc,* se creó la carpeta switches ejecutando el siguiente comando:

```
mkdir switches
```

3. Dentro de la carpeta switches, se crearon los archivos de los host. Nagios ofrece archivos de ejemplo para agregar hosts, para esta configuración se hizo uso del archivo *switch.cfg* como plantilla para agregar los equipos.

   El archivo *switch.cfg* se encuentra en la ruta */usr/local/nagios/etc/objects/*.

```
root@watcher:~# ls /usr/local/nagios/etc/switches/
equipos_CheckPoint.cfg  equipos_Forti.cfg  equipos_PaloAlto.cfg  equipos_TrendMicro.cfg
```

Ilustración 18. Lista de los archivos que contienen hosts.

Archivos de host.

Para esta configuración, como ya se mencionó anteriormente, se dividieron los host por tecnologías, por lo que se crearon 4 archivos, *equipos_CheckPoint.cfg, equipos_PaloAlto.cfg, equipos_TrendMicro.cfg, equipos_Forti.cfg.*

Lo primero que se ubica dentro de cada archivo es, la declaración de los host, posteriormente los hostgroup y por último los servicios que se monitorean para esa tecnología.

A continuación, se presenta el contenido del archivo *equipos_Forti.cfg* y se explica su contenido.

Como se aprecia en la ilustración 19, los host fueron separados por cliente, *"define host"* son las palabras reservadas para declarar un host, y todos los parámetros que se encuentrén dentro de las llaves definen a dicho host.

El campo *"use"* indica qué plantilla se usará, en este caso se usa *"generic-switch"*, *"host_name"* indica el nombre del host que se registra, este nombre puede ser determinado por el usuario, *"address"* indica la dirección IP del equipo, "hostgroups" indica a que grupo o grupos de host pertenece dicho equipo, si pertenece a varios grupos, se deben separar por comas, *"_SNMP_COMMUNITY"* indica el nombre de la comunidad SNMP que se configuró en el equipo, *"max_check_attempts"* define la cantidad de veces que Nagios intentará ejecutar el comando si el host devuelve un estado diferente a *"OK"*, *"notifications_enabled"* se usa para determinar si las notificaciones para el host están habilitadas o no, *"check_interval"* define el tiempo entre las comprobaciones regulares al host, *"retry_interval"* define el tiempo que se debe esperar antes de volver a verificar el estado del host, *"contacts"* indica el nombre de los contactos a los que se les debe enviar notificaciones en caso de obtener un estado de alerta, *"notification_options"* indica qué estados del host ameritan el envio de notificaciones, *"notification_interval"* indica el tiempo que se debe esperar antes de volver a enviar nuevamente una notificación, *"notification_period"* indica el periodo de tiempo durante el cual se pueden enviar notifiaciones del host.

```
################################################################################
#                                HOST FORTI                                    #
################################################################################
#############
#HOST PRUEBA#
#############
define host{
    use                             generic-switch
    host_name                       Lab_Nagios_Forti
    address                         10.11.11.54
    hostgroups                      equipos_Forti
    _SNMP_COMMUNITY                 Pruebas_SNMP_v2
    max_check_attempts              3
    notifications_enabled           1
    check_interval                  1
    retry_interval                  1
    contacts                        FORTIadmin
    notification_options            d,u,r
    notification_interval           5
    notification_period             24x7
}
############
#HOST A**BC#
############
define host {
    use                             generic-switch
    host_name                       ********
    address                         ***.fortiddns.com
    hostgroups                      equipos_Forti, ***
    _SNMP_COMMUNITY                 ***
    max_check_attempts              3
    notifications_enabled           1
    check_interval                  1
    retry_interval                  1
    contacts                        FORTIadmin
    notification_options            d,u,r
    notification_interval           5
    notification_period             24x7
}
```

Ilustración 19. Definición de host dentro del archivo equipos_Forti.cfg.

En la ilustración 20 se observa la declaración de algunos de los host groups. Para declarar un host group se usan las palabras reservadas *"define hostgroup"*, *"hostgroup_name"* indica el nombre del grupo, *"alias"* indica un nombre corto para el hostgroup y *"members"* enlista a los host que pertenecen a ese grupo, los host se separan por comas.

```
###############################################################################
#                         HOST GROUP DEFINITIONS                              #
###############################################################################
# Create a new hostgroup for switches

define hostgroup {
    hostgroup_name                  equipos_Forti ; The name of the hostgroup
    alias                           equipos_Forti ; Long name of the group
}

define hostgroup{
    hostgroup_name                  ***
    alias                           ***_HOSTS
    members                         ***_MON
    }

define hostgroup{
    hostgroup_name                  ******
    alias                           ******_HOSTS
    members                         ******_EMS, ******_FAZ, ******_CORPORATIVO
    }
```

Ilustración 20. Definición de hostgroups dentro del archivo equipos_Forti.cfg

Los servicios básicos que se monitorearon para los equipos son:

- Ping.
- Uptime.
- Estado del CPU.
- Estado de la memoria.
- Número de sesiones.
- VPN.
- Estado de las interfaces.

Para definir un servicio, se usa *"use generic-service"*, para indicar que la plantilla que usará es la de un servicio, *"host_name"* enlista a los host a los que se les monitoreará dicho servicio, los host deben separarse con comas, *"service_description"* indica un nombre con el que se pueda identificar dicho servicio, *"check_command"* indica el nombre del servicio y ciertos parámetros adicionales que dependen de la definición del comando y los requerimientos del usuario, el nombre debe coincidir con el nombre declarado en *commands.cfg*, *"check_interval"* define el tiempo entre las comprobaciones regulares al host, *"retry_interval"* define el tiempo que se debe esperar antes de volver a verificar el estado del host, *"notifications_enabled"* se usa para determinar si las notificaciones para el host están habilitadas o no, *"notification_period"* indica el periodo de tiempo

durante el cual se pueden enviar notifiaciones del host, *"notification_interval"* indica el tiempo que se debe esperar antes de volver a enviar nuevamente una notificación, *"contacts_groups"* indica los groupcontacts a los que se les debe enviar notificaciones en caso de obtener un estado de alerta, *"max_check_attempts"*, define la cantidad de veces que Nagios intentará el comando de verificación del host si devuelve un estado diferente a "OK", "notification_options" indica qué estados del host ameritan el envio de notificaciones.

Anteriormente se mencionó que *"check_command"* indica el nombre del servicio a usar, además indica parámetros adicionales para el funcionamiento de dicho comando, a continuación se explica brevemente el parámetro *"check_command"* de cada servicio.

```
###############################################################################
#                             SERVICE DEFINITIONS                             #
###############################################################################

# PING
define service {
    use                             generic-service
    host_name                       Lab_Nagios_Forti, *****, *****, *****
    service_description             PING
    check_command                   check_ping!200.0,20%!600.0,60%
    check_interval                  2
    retry_interval                  1
    notifications_enabled           1
    notification_period             24x7
    notification_interval           5
    contact_groups                  FORTIadmins
    max_check_attempts              1
    notification_options             w,u,c,r
}

# UPTIME
define service  {
    use                             generic-service
    host_name                       Lab_Nagios_Forti
    service_description             UPTIME
    check_interval                  2
    retry_interval                  1
    notifications_enabled           1
    notification_period             24x7
    notification_interval           5
    contact_groups                  FORTIadmins
    max_check_attempts              1
    notification_options             w,u,c,r
    check_command                   CHKP_UPTIME
}
```

Ilustración 21. Definición de los servicios ping y uptime dentro del archivo equipos_Forti.cfg.

Observe la ilustración 22, dentro del servicio PING, el parámetro *check_command* hace uso del comando *check_ping* de la siguiente forma:

```
check_ping!200.0,20%!600.0,60%
```

Nagios ofrece la declaración del comando *check_ping* de la siguiente forma:

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

Se observa en el campo *command_line* que se requieren 4 argumentos, el correspondiente a *"-H"* es la dirección del host, para *"-w"* es el valor que genera una alerta *"WARNING"*, para *"-c"* es el valor que dispara una alerta *"CRITICAL"* y para *"-p"* es el número de paquetes que se enviarán.

También se observa que en el parámetro *"-H"* se hace uso de la macro *$HOSTADDRESS$* por lo que, este valor se obtiene de la declaración de cada host y para el parámetro *"-p"* se declara el valor 5, en el caso de los parámetros *"-w"* y *"-c"* se hace uso de macros qué permiten que el usuario ingrese valores de forma manual, por lo que solo esos dos argumentos recibirán una entrada.
Retomando la declaración del servicio PING:

```
check_ping!200.0,20%!600.0,60%
```

Se observa el uso del símbolo *"!"*, cuya función es separar parámetros, *200.0,20%* es el argumento correspondiente al parámetro *w*, y *600.0,60%* es el argumento correspondiente al parámetro *c*. Por lo tanto, cada que se pierda el 20% de los paquetes enviados, el estado que devuelve el servicio *PING* es *"WARNING"* y cada que se pierda el 60% de los paquetes enviados, el estado que devuelve el servicio PING es *"CRITICAL"*. Ya no fue necesario declarar los parámetros *"-H"* ni *"-p"* porque desde la definición del comando, estos parámetros ya tienen un valor o una macro que les asigna un valor.

En el caso del servicio UPTIME, CPU, MEMORY SESSIONS y VPN, no fue necesario agregar parámetros adicionales ya que en la declaración del comando, se hizo uso de macros que obtienen los valores para los parámetros.

```
# CPU
define service {
    use                             generic-service
    host_name                       Lab_Nagios_Forti
    service_description             CPU
    check_command                   check_fortigate_cpu
    check_interval                  2
    retry_interval                  1
    notifications_enabled           1
    notification_period             24x7
    notification_interval           5
    max_check_attempts              1
    notification_options            w,u,c,r
    contact_groups                  FORTIadmins
}
#MEMORIA
define service {
    use                             generic-service
    host_name                       Lab_Nagios_Forti, *****, *****
    service_description             MEMORY
    check_command                   check_fortigate_mem
    check_interval                  2
    retry_interval                  1
    notifications_enabled           1
    notification_period             24x7
    notification_interval           5
    contact_groups                  FORTIadmins
    max_check_attempts              1
    notification_options            w,u,c,r
}
```

Ilustración 22. Definición de servicios CPU y MEMORY dentro del archivo equipos_Forti.cfg.

```
#SESSIONS
define service {
    use                             generic-service
    host_name                       Lab_Nagios_Forti, *****, *****
    service_description             SES
    check_command                   check_fortigate_ses
    check_interval                  2
    retry_interval                  1
    notifications_enabled           1
    notification_period             24x7
    notification_interval           5
    max_check_attempts              1
    notification_options            w,u,c,r
    contact_groups                  FORTIadmins
}
#VPN
define service {
    use                             generic-service
    host_name                       Lab_Nagios_Forti
    service_description             VPN
    check_command                   check_fortigate_vpn
    check_interval                   2
    retry_interval                  1
    notifications_enabled           1
    notification_period             24x7
    notification_interval           5
    max_check_attempts              1
    notification_options            w,u,c,r
    contact_groups                  FORTIadmins
}
```

Ilustración 23. Definición de servicios sesiones y VPN dentro del archivo equipos_Forti.cfg.

Como se observa en la ilustración 24, para el caso de los servicios que monitorean las distintas interfaces de los dispositivos, fue necesario indicar la versión de SNMP que se usa y el número de interfaz a monitorear.

```
define service {
    use                             generic-service
    host_name                       Lab_Nagios_Forti
    service_description             INTERFACE 1
    check_command                   check_traffic_interfaces!2c!1
    check_interval                  2
    retry_interval                  1
    notifications_enabled           1
    notification_period             24x7
    notification_interval           5
    max_check_attempts              1
    notification_options            w,u,c,r
    contact_groups                  FORTIadmins
}

define service {
    use                             generic-service
    host_name                       Lab_Nagios_Forti, *****, *****, *****
    service_description             INTERFACE 12
    check_command                   check_traffic_interfaces!2c!12
    check_interval                  2
    retry_interval                  1
    notifications_enabled           1
    notification_period             24x7
    max_check_attempts              1
    notification_options            w,u,c,r
    contact_groups                  FORTIadmins
    notification_interval           5
}
```

Ilustración 24. Definición de servicios de monitoreo de interfaces dentro del archivo equipos_Forti.cfg.

Notificaciones.

El envío de notificaciones, en caso de que se presente alguna anomalía en el estado de los equipos, es esencial por lo que, en esta configuración se configuró el envío de alertas por Telegram y por correo electrónico en caso de fallas o estados críticos.

1. Se agregaron los datos de los contactos que recibirán las notificaciones, para ello, se modificó el archivo contacts.cfg ubicado en la ruta */usr/local/nagios/etc/objects/* y se agregaron los contactos y grupos necesarios. Para esta configuración, se dividieron los grupos de contactos por tecnologías.

```
###############################################################################
# CONTACT GROUPS
###############################################################################

define contactgroup {
    contactgroup_name       CHKadmins
    alias                   Check Point Nagios Administrators
    members                 CHKadmin
}

define contactgroup {
    contactgroup_name       FORTIadmins
    alias                   Fortinet Nagios Administrators
    members                 FORTIadmin
}
```

Ilustración 25.Declaración de los contactgroups en contacts.cfg.

```
define contact {
   contact_name                    FORTIadmin
   use                             generic-contact
   alias                           Fortigate admin
   email                           fernanda.prueba@mail.com
   service_notification_period     24x7
   host_notification_period        24x7
   host_notification_options       d,u,r,f,s
   service_notification_options    w,u,c,r,f,s
   service_notification_commands   notify-service-by-email,notify-service-by-
telegram_F
   host_notification_commands      notify-host-by-email,notify-host-by-
telegram_F
}

define contact {
   contact_name                    CHKadmin
   use                             generic-contact
   alias                           Check Point Admin
   email                           fernanda.prueba@mail.com
   service_notification_period     24x7
   host_notification_period        24x7
   service_notification_options    w,u,c,r,f,s
   host_notification_options       d,u,r,f,s
   service_notification_commands   notify-service-by-email,notify-service-by-
telegram_CHK
   host_notification_commands      notify-host-by-email,notify-host-by-
telegram_CHK
}
```

Ilustración 26. Declaración de los contactos, y sus parámetros.

2. Para la declaración de los contactos, se usó la plantilla de contactos contenida en el archivo *templates.cfg* ubicada en */usr/local/nagios/etc/objects/*.

3. Fue necesario declarar los campos descritos a continuación:
    - Para indicar que las notificaciones tanto de los host como de los servicios serán enviadas las 24 horas del día, los 7 días de la semana se usó:

```
service_notification_period     24X7
host_notification_period         24X7
```

    - Para indicar qué estados de los host y de los servicios, ameritan una notificación se hace uso de:

```
service_notification_options     w,u,c,r,f,s
```

      Las letras a la derecha de *service_notification_options* indican que si el estado de un servicio es WARNING, UNKNOWN, CRITICAL, RECOVERY, o FLAPPING, se enviará una notificación.

```
host_notification_options          d,u,r,f,s
```

      Las letras a la derecha de *host_notification_options* indican que si el estado de un host es DOWN, UNREACHABLE, RECOVERY o FLAPPING, se enviará una notificación.
    - Fue necesario indicar qué comandos se usarán para el envío de notificaciones, para ello se declaró:

```
service_notification_commands     notify-service-by-email, notify-
service-by-telegram
host_notification_commands        notify-host-by-email, notify-host-
by-telegram
```

Los comandos anteriores, notificarán al contacto por email o por Telegram si algún servicio o host, presenta alguna anomalía, a continuación se explica, como se desarrollaron dichos comandos.

Telegram.

Para el envío de notificaciones por Telegram:

1. Dentro de la aplicación de Telegram, en cualquier conversación se envió:

```
@BotFather
```

2. El mensaje, apareció como un vínculo.

Ilustración 27. Mensaje @BotFather en Telegram.

3. Se abrió una conversación en la que se desplegó un menú con opciones:

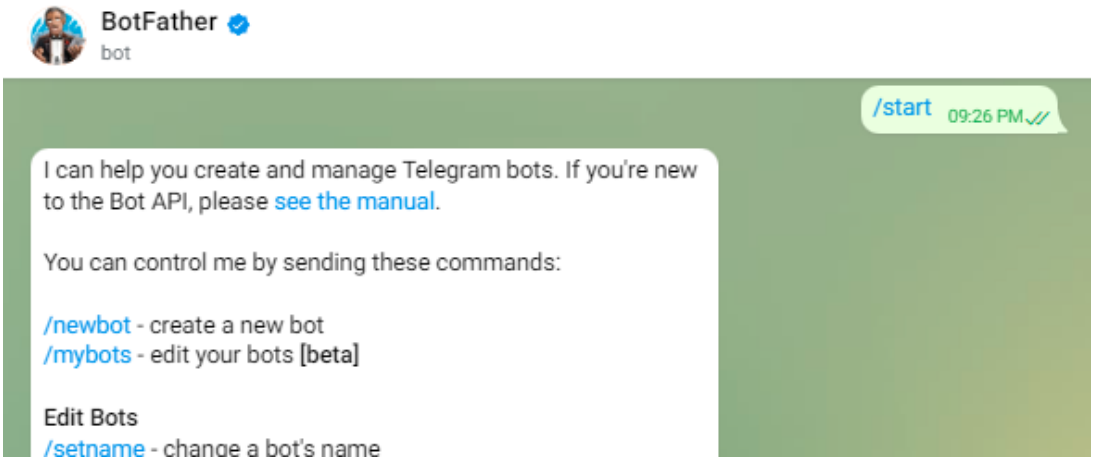

Ilustración 28. Chat con @BotFather.

4. Fue necesario crear un bot, para ello, se escribió `/newbot` y después se declaró el nombre del Bot, dicho nombre debe estar disponible. El BotFather desplegó un mensaje con el token del Bot para acceder desde la API de HTTP.

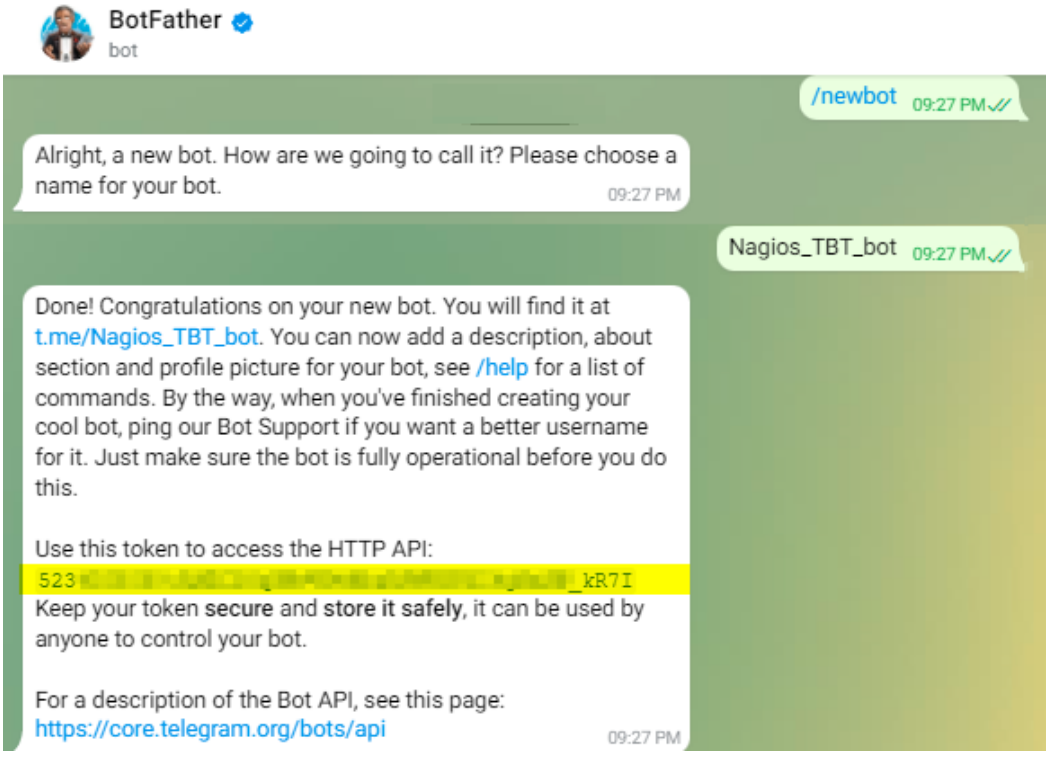

Ilustración 29. Chat con @BotFather, obtención del token.

5. Se deshabilitó la privacidad del Bot, para ello, se escribió */setprivacy*, se seleccionó el Bot, y se deshabilitó la privacidad del Bot escribiendo *"Disable"*.

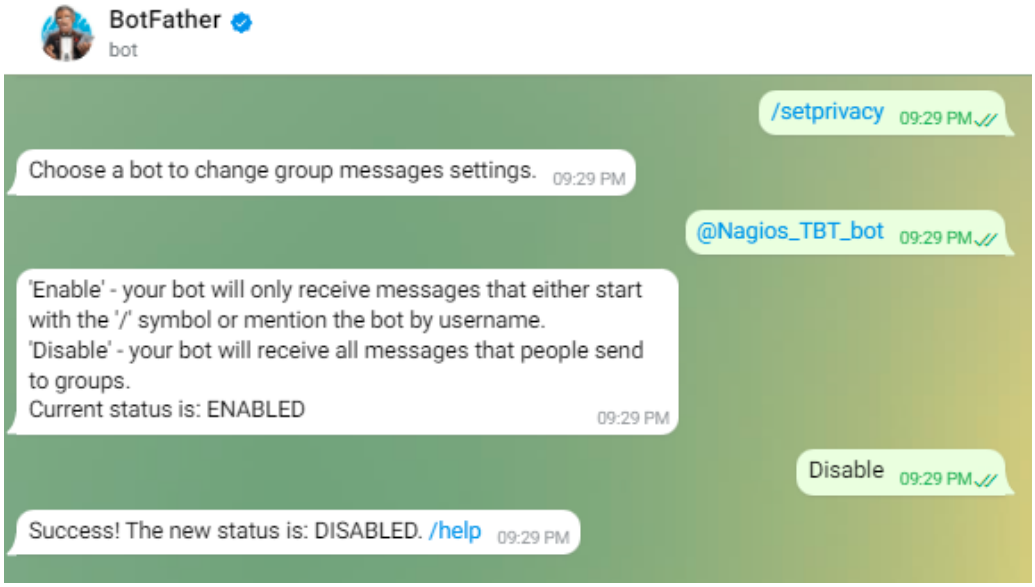

Ilustración 30. Chat con @BotFather deshabilitar privacidad del Bot.

6. Se creó un grupo en Telegram, y se agregó al Bot como miembro.
7. Desde la consola de la máquina, se escribió:

```
curl -Lk -i -X GET https://api.telegram.org/botACCESSTOKEN-GIVEN-FROM-BOTFATHER/getUpdates
```

Donde se sustituyó *ACCESSTOKEN-GIVEN-FROM-BOTFATHER* por el token obtenido al momento de crear al Bot.

8. Se desplegó un mensaje en el que se identificó la frase *"chat":{"id":-xxxxxxxxx….* y se guardó el id del chat.

```
"message":{"message_id":14,"from":{"id":300920731,"is_bot":false,"first_name":"Your","last_name":"Name","language_code":"en-US"},"chat":{"id":-123456789,"title" :"My Telegram group","type":"supergroup"},"date":1520901132,"text":"Helloworld! "}}]}
```

9. Dentro del archivo *commands.cfg* se agregaron los siguientes comandos:

```
define command{
    command_name    notify-host-by-telegram
    command_line    curl -k -L --data chat_id=REPLACEME1 --data-urlencode "text=***** Nagios ***** Notification Type: $NOTIFICATIONTYPE$ Host: $HOSTNAME$ State: $HOSTSTATE$ Address: $HOSTADDRESS$ Info: $HOSTOUTPUT$ Date/Time: $LONGDATETIME$" "https://api.telegram.org/botREPLACEME2/sendMessage"
}
```

```
define command{
    command_name    notify-service-by-telegram
    command_line    curl -k -L --data chat_id=-REPLACEME1 --data-
urlencode "text=***** Nagios ***** Notification Type: $NOTIFICATIONTYPE$
Service: $SERVICEDESC$ Host: $HOSTALIAS$ Address: $HOSTADDRESS$ State:
$SERVICESTATE$ Date/Time: $LONGDATETIME$ Additional Info:
$SERVICEOUTPUT$" "https://api.telegram.org/botREPLACEME2/sendMessage"
        }
```

Donde se sustituyó *REPLACEME1* por el id del chat y *REPLACEME2* por el token del Bot.

Al final los comandos se verán así:

```
define command {

    command_name    notify-host-by-telegram_
    command_line    curl -k -L --data chat_id=-******* --data text="***** Nagios
*****%0ANotification Type: $NOTIFICATIONTYPE$%0AHost: $HOSTNAME$%0AState:
$HOSTSTATE$%0AAddress: $HOSTADDRESS$%0AInfo: $HOSTOUTPUT$%0ADate/Time:
$LONGDATETIME$" https://api.telegram.org/bot523**********************/sendMessage
}


define command {

    command_name    notify-service-by-telegram_
    command_line    curl -k -L --data chat_id=-******* --data text="***** Nagios
*****%0ANotification Type: $NOTIFICATIONTYPE$%0AService: $SERVICEDESC$%0AHost:
$HOSTALIAS$%0AAddress: $HOSTADDRESS$%0AState: $SERVICESTATE$%0ADate/Time:
$LONGDATETIME$%0AAdditional   Info: $SERVICEOUTPUT$" https://api.telegram.org/
bot523**********************/sendMessage
}
```

Ilustración 31. Declaración de comandos para envío de notificaciones por Telegram.

10. Para esta configuración, se enviarán las notificaciones a dos grupos distintos de Telegram, por lo que fue necesario declarar los comandos anteriormente descritos, en dos ocasiones, el chat id dentro de cada comando cambió pero el Bot es el mismo.

Correo electrónico.

Nagios ofrece comandos predeterminados para el envío de notificaciones por correo electrónico, para ello se hace uso de la aplicación mail, pero fue necesario instalar un mailx que permite enviar correos usando autenticación del servidor SMTP usado.

1. Instalación de mailx:

```
yum install mailx
```

2. Fue necesario realizar modificaciones a la configuración del proveedor del servicio del correo electrónico, que en este caso es Google, por lo que, dentro de la configuración de correo electrónico, se habilitó la verificación de dos pasos y se creó una contraseña para aplicación.

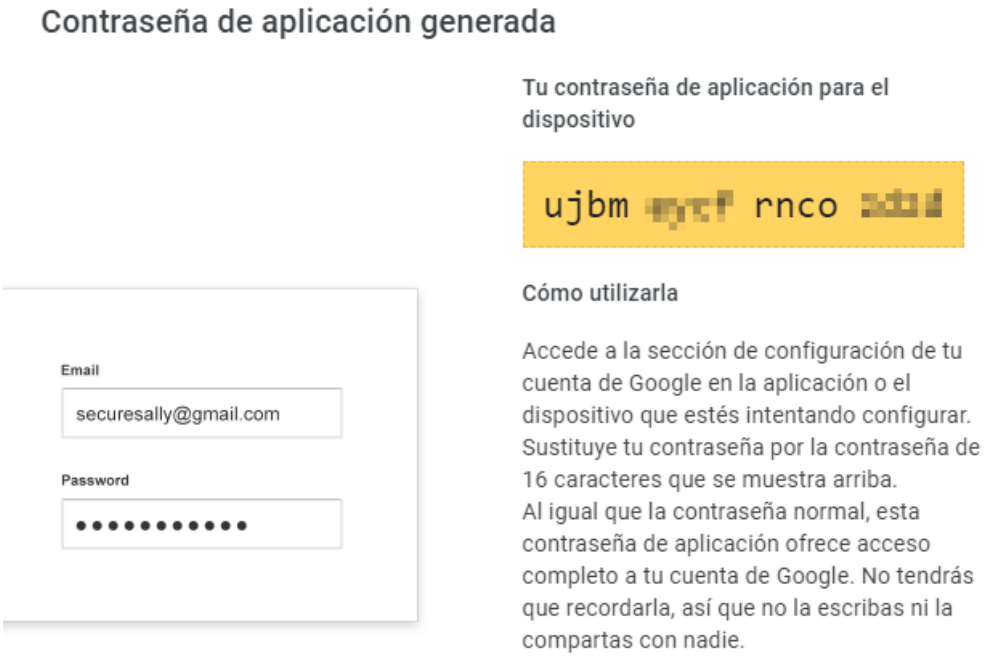

Ilustración 32. Contraseña de aplicación generada.

3. Se modificó nuevamente el archivo *commands.cfg* sustituyendo los comandos existentes para enviar emails por los siguientes:

```
define command {
    command_name     notify-host-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios*****
\n\nNotification Type: $NOTIFICATIONTYPE$\nHost: $HOSTNAME$\nState:
$HOSTSTATE$\nAddress: $HOSTADDRESS$\nInfo: $HOSTOUTPUT$\n\nDate/Time:
$LONGDATETIME$\n" | /bin/mailx -r "remitente@mail.com" -s "**
$NOTIFICATIONTYPE$ Host Alert: $HOSTNAME$ is $HOSTSTATE$ **" -S
smtp="smtp.gmail.com:587" -S smtp-use-starttls -S smtp-auth=login -S smtp-
auth-user="remitente@mail.com" -S smtp-auth-
password="contraseñaDeAplicación" -S ssl-verify=ignore $CONTACTEMAIL$
}
```

```
define command {
    command_name    notify-service-by-email
    command_line    /usr/bin/printf "%b" "***** Nagios
*****\n\nNotification Type: $NOTIFICATIONTYPE$\n\nService:
$SERVICEDESC$\nHost: $HOSTALIAS$\nAddress: $HOSTADDRESS$\nState:
$SERVICESTATE$\n\nDate/Time: $LONGDATETIME$\n\nAdditional
Info:\n\n$SERVICEOUTPUT$\n" | /bin/mailx -r ="remitente@mail.com" -s "**
$NOTIFICATIONTYPE$ Service Alert: $HOSTALIAS$/$SERVICEDESC$ is
$SERVICESTATE$ **" -S smtp="smtp.gmail.com:587" -S smtp-use-starttls -S
smtp-auth=login -S smtp-auth-user="remitente@mail.com" -S smtp-auth-
password="contraseñaDeAplicación"-S ssl-verify=ignore $CONTACTEMAIL$

}
```

Donde se sustituyó *"remitente@mail.com"* por el correo desde el que se enviarán los correos y *"contraseñaDeAplicación"* se sustituyó por la contraseña de aplicación generada.

Resultados

Los objetivos propuestos para el proyecto se convirtieron en metas alcanzadas, obteniendo como resultado los módulos de monitoreo, gráficas y reportes integrados, permitiendo optimizar el monitoreo, y la respuesta a incidentes a través de las notificaciones en tiempo real, además de desarrollar una herramienta que permitiera obtener reportes del estado de los equipos de una manera sencilla y eficaz.

Por motivos de políticas de privacidad dentro de la empresa, se implementaron dos equipos de prueba, uno de la marca Fortinet y otro Check Point para así corroborar el funcionamiento del sistema de monitoreo.

Las imágenes de la 33 a la 37 muestran las acciones permitidas en los módulos desarrollados.

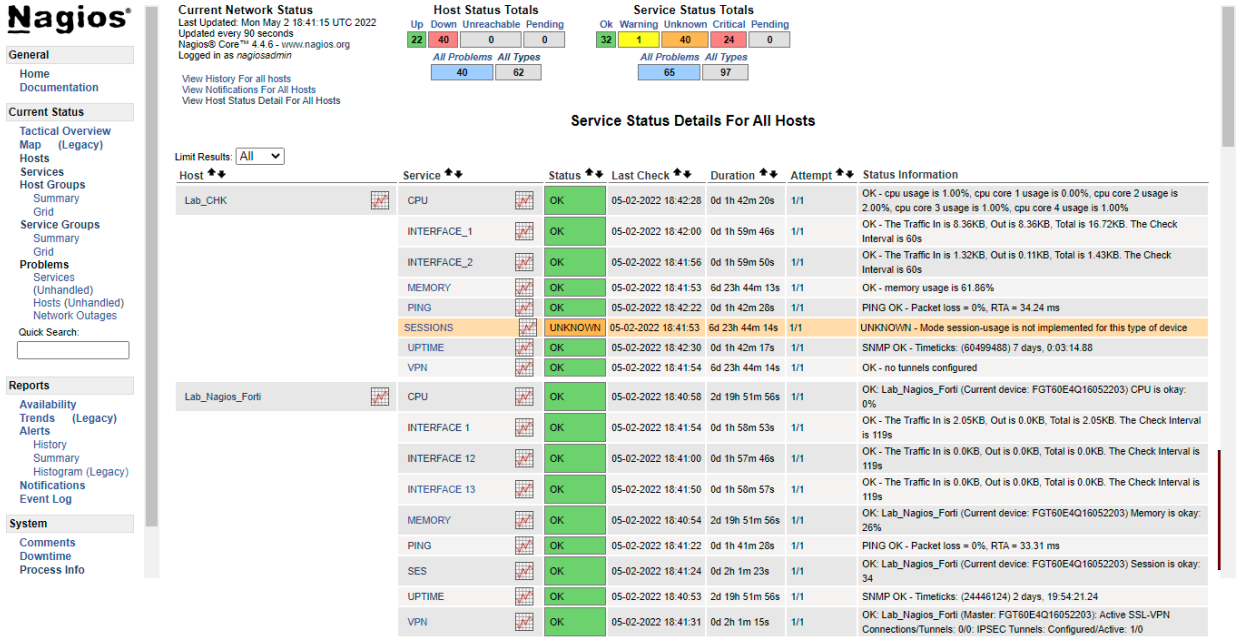

Ilustración 33. Interfaz web de Nagios y despliegue de host monitoreados.

La ilustración 34 muestra las gráficas de los servicios monitoreados en un host, se pueden incluso desplegar gráficas de diferentes host en forma simultánea. Este módulo, permite al usuario tener total libertad sobre como desea realizar el despliegue de los datos, permitiendo agregar gráficos de diferentes host, servicios o rangos de tiempo.

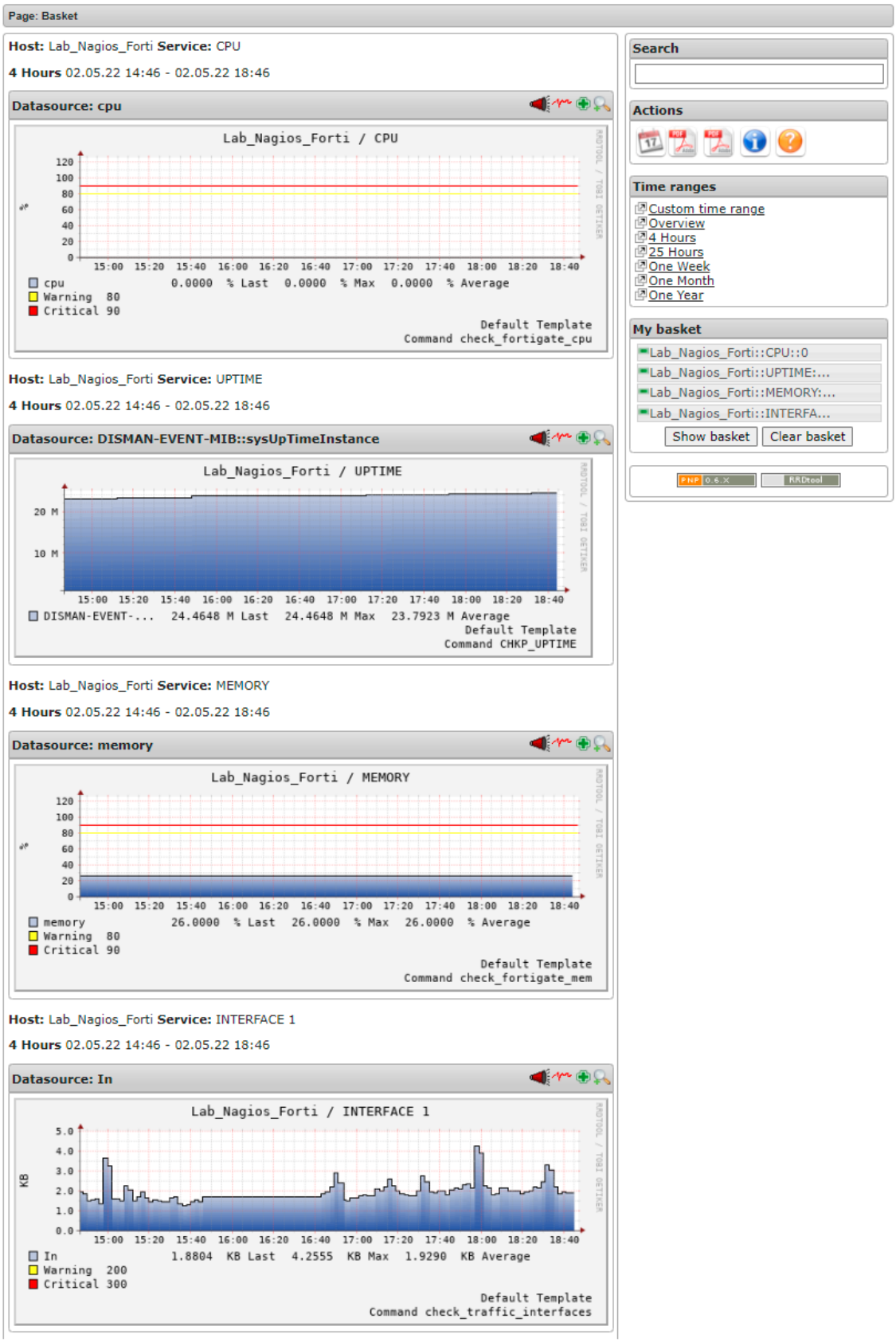

Ilustración 34. Interfaz web de PNP4Nagios y despliegue de las gráficas.

En la ilustración 35 puede observar que se generó un reporte en formato PDF con los datos obtenidos del monitoreo del host.

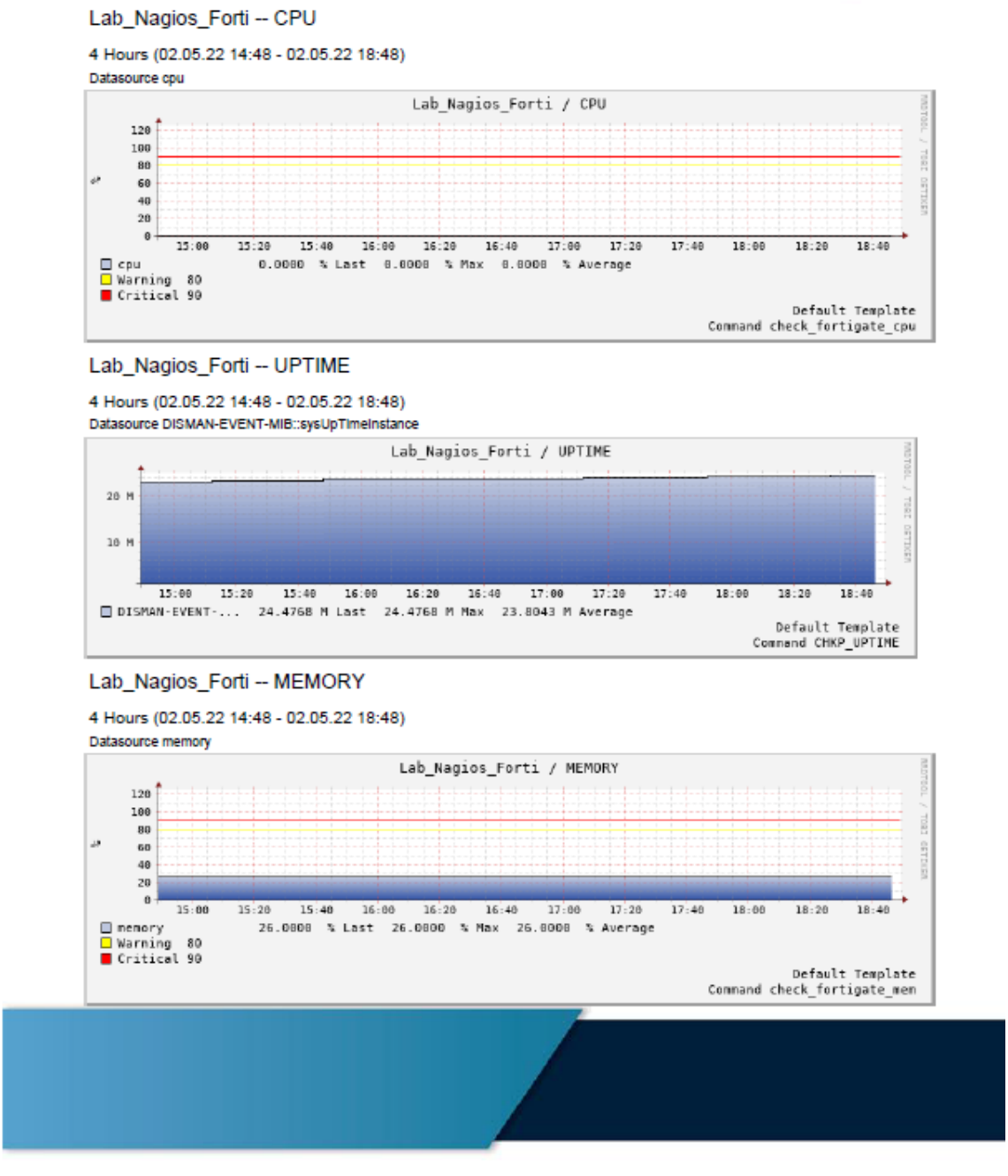

Ilustración 35. Reporte PDF generado con los servicios del host.

En las ilustraciones 36 y 37 se muestra el despliegue alertas en Telegram y en el correo electrónico cuando un host o servicio presentan un diferente a *"OK"*.

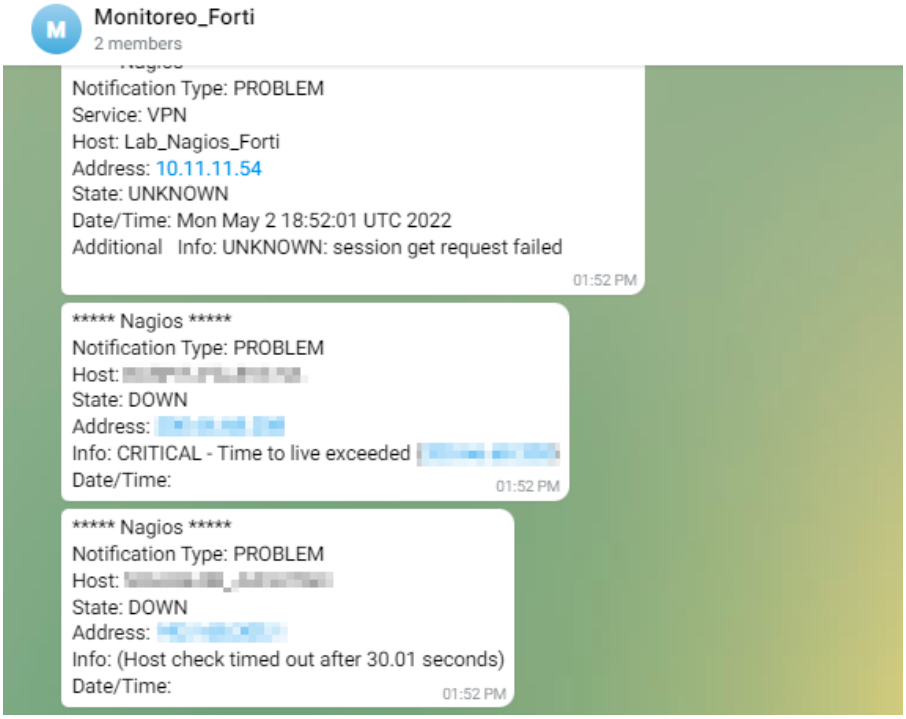

Ilustración 36. Alertas a través de Telegram.

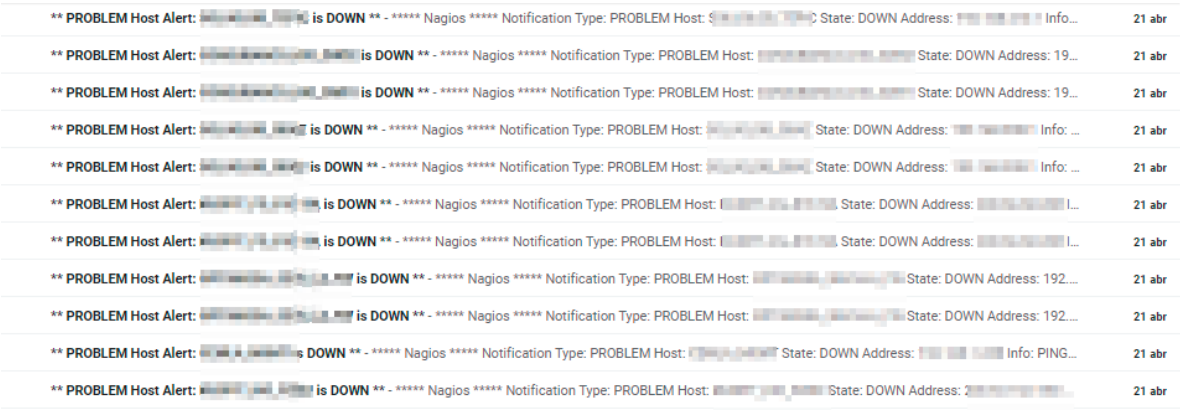

Ilustración 37. Alertas a través del correo electrónico.

Puesto que, la aplicación cumple con los objetivos propuestos, es posible mencionar que aún es plausible realizar mejoras y agregar funcionalidades no previstas en los objetivos del proyecto.

Análisis y discusión de resultados

Con el monitoreo de los dispositivos de la red, se puede observar de forma remota el comportamiento de los equipos, en caso de que se presente alguna falla o anomalía, se puede actuar de forma oportuna y casi inmediata, evitando afectaciones totales o parciales dentro de la operación de la infraestructura de los clientes.

Este sistema de monitoreo se tendrá que mejorar a futuro para integrar más dispositivos y características a monitorear.

Conclusiones

Se logró obtener un sistema que permite recopilar, administrar y analizar los datos del estado y funcionamiento de los dispositivos y servicios monitoreados; en las pruebas realizadas con los equipos de prueba, no se presentaron problemas en la operación al realizar las tareas solicitadas. Queda pendiente la configuración de los equipos que la empresa monitorea.

Previo a la instalación y configuración de la herramienta de monitoreo en un servidor de producción, se hicieron todas las configuraciones en una máquina virtual de prueba para evitar, a medida de lo posible que se generen afectaciones en la operación de la empresa y sus dispositivos, entre los posibles problemas, se encuentra el uso de espacio de almacenamiento como una limitante, ya que la herramienta almacena los datos de los monitoreos en archivos que aumentan su tamaño continuamente.

Sería recomendable hacer un análisis de la aplicación para complementar el desarrollo de esta en ciertos puntos de interés para mejorar el funcionamiento o incluso agregar funcionalidades.

Bibliografía

[1] CISCO. (2020, 28 octubre). ¿Qué es el monitoreo de red? https://www.cisco.com/c/es_mx/solutions/automation/what-is-network-monitoring.html

[2] Carmona Lemus, J., & Ponce Leyva, G. (1991). Monitoreo del funcionamiento de motores de C.A. empleados en procesos industriales mediante un sistema en red centralizado a una computadora personal. México, D.F. http://espartaco.azc.uam.mx/tesis/X13377.pdf

[3] Alemán Negrete, J. C. (2011). Sistema de Monitoreo de Alarmas de Emergencia. México, D.F. http://espartaco.azc.uam.mx/tesis/x17302.pdf

[4] Bazán Rosales, L., Montalvo Bazán, J., & Montes Sánchez, D. (2005). Sistema de monitoreo y registro de temperaturas para incubadoras. México, D.F. http://espartaco.azc.uam.mx/tesis/x16557.pdf

[5] Bayas Villagómez, J. I. (2015, julio). Servidor de control de dispositivos y servicios mediante el protocolo SNMP para la red de datos en CELEC.E.P. unidad de negocio Hydragoyán.
https://repositorio.uta.edu.ec/bitstream/123456789/13063/1/Tesis_t1035ec.pdf

[6] Orrala, B. E. D. R. (2016, 24 febrero). Repositorio Universidad Estatal Península de Santa Elena: Implementación de monitoreo de red utilizando los Protocolos icmp y snmp. https://repositorio.upse.edu.ec/handle/46000/2583

[7] Arellano, A. (2012, 1 agosto). DSpace ESPOCH.: Análisis del protocolo SNMPv3 para el desarrollo de un prototipo de monitoreo de red segura. http://dspace.espoch.edu.ec/handle/123456789/2037

[8] "¿Qué es SNMP? | Protocolo SNMP Monitorización". ManageEngine - IT Operations and Service Management Software. https://manageengine.com/es/network-monitoring/what-is-snmp.html

[9] "¿Qué es el monitoreo de red?" Cisco.
https://www.cisco.com/c/es_mx/solutions/automation/what-is-network-monitoring.html

[10] "¿QUE ES ZABBIX? Herramienta de monitoreo de redes". Quasar Software.
https://quasarbi.com/ZABBIX.html

[11] "¿Que es nagios?" NORTH NETWORKS | Seguridad para Infraestructuras de TI.
https://www.north-networks.com/que-es-nagios/

Entregables

Un CD que contiene el informe del proyecto de integración en formato PDF.