Nagios plugin. HP ILO.

Vamos a ver como monitorizar un servidor HP a través de su interface de gestión ILO. El plugin correspondiente nos avisará en caso de que exista una problema de Hardware en el servidor (Temperaturas, fuentes de alimentación, memoria, micros,…)

Para chequear la salud Hardware de servidores HP tenemos básicamente dos opciones:

1. Realizarlo a través del puerto de gestión ILO.
2. Realizarlo a través de SNMP para lo que necesitamos que el Sistema Operativo del Servidor tenga instalado el agente de HP Insight Manager que es el que nos añade las librerías necesarias para disponer de esos datos por SNMP.

En este articulo veremos la primera opción, mediante ILO. Usaremos el plugin disponible en NagiosExchange: check_ilo2_health. Mirar bien los comentarios en la página ya que hay en estos un enlace a una versión del plugín más actualizada. Este plugin también nos funcionará para las ilo3 y parece que para ilo4.

Lo bajamos y copiamos nuestro directorio de plugins y lo hacemos ejecutable. En nuestro caso lo probamos en un servidor Debian y fue necesario previamente tener instalado algunos módulos de Perl.

```
apt-get install libnet-smtp-ssl-perl libxml-simple-perl
```

Lo probamos en la línea de comandos contra el servidor, con usuario Administrator p.e.:

```
./check_ilo2_health.pl -H 192.0.0.10 -u Administrator –p Password -n -3
```

OJO. Si es ILO3/4 añadimos el Flag “-3”

Si queremos ver la salida de todo lo que chequea usaremos el flag de verbose -v. Nos podemos hacer una idea de esta forma de todo lo que chequea. Mirando la ayuda podemos ver que tiene varias opciones activables para chequear también discos, fuente redundante, datos de rendimiento para las gráficas,…

Podemos configurar el check_command para recibir los parámetros habitualmente o podemos hacer uso de macros variables a medida (Custom Variable Macros) para simplificar el acceso a todas las ILOs. En este caso, para aportar algo más , lo haremos de esa manera. En otro artículo hablaremos más a fondo del tema.

Nuestros commands de ambas formas (con y sin macros variables a medida) serian básicamente los siguientes. Usaremos en los ejemplos el segundo:

```
# CHECK_ILO
 define command {
 command_name check_ilo
 command_line $USER1$/check_ilo2_health.pl -u $USER10$ -p $USER11$ -H $HOSTADDRESS$ -n
}
# CHECK_ILO, con macros a medida
 define command {
 command_name check_ilo_mod
 command_line $USER1$/check_ilo2_health.pl -u $_HOSTILO_USER$ -p $_HOSTILO_PASS$ -H
$_HOSTILO_ADDRESS$ -n
}
```

¿Por qué usar "custom variable macros"?. En este caso nos encontramos con que tenemos un servidor que monitorizar por dos direcciones ips con servicios diferentes (ip S.O. e ip de la ILO). Tenemos la opción de crear un Host independiente para la ILO con su dirección IP y listo, sin complicaciones. Para este caso podemos usar como nombre de Host para la ILO el mismo que para el S.O. pero añadiéndole al final "-ILO", para que aparezcan seguidos. La otra opción, usando estas macros, es definir directamente el chequeo de la ILO como un servicio más de nuestro servidor. También se podía haber hecho a mano pero de esta forma pienso ganamos en claridad y orden. Todos nuestros hosts con ILOs los podemos definir de la misma forma.

```
define host{
    use generic-host
    host_name virtual3
    alias virtual3
    address 192.168.2.12
    _ILO_ADDRESS 192.168.1.12
    _ILO_USER Administrator
    _ILO_PASS XXXXXXX
}
```

Y la definición del servicio sería p.e. para un host

```
# Servicio - Chequeo de estado equipos HP a traves de ILO -
define service {
   use generic-service
   host_name virtual3
   #hostgroup_name HPILO-Servers
   service_description EstadoHardwareHP_ILO2
   check_command check_ilo_mod
}
```

Siempre será mejor crear un grupo de hosts que contenga todos los que tienen ILO. El resultado sería el siguiente:

EstadoHardwareHP_ILO	OK	2013-04-17 11:52:07	0d 0h 0m 27s	1/3	ILO2_HEALTH OK - No faults detected
VMware CPU Usage11	OK	2013-04-17 11:44:47	0d 23h 37m 47s	1/3	CHECK_VMWARE_API.PL OK - cpu usage=22.56 %
VMware IO Reads	OK	2013-04-17 11:50:39	5d 21h 1m 55s	1/3	CHECK_VMWARE_API.PL OK - io read latency=0 ms
VMware IO Write	OK	2013-04-17 11:45:33	0d 3h 47m 1s	1/3	CHECK_VMWARE_API.PL OK - io write latency=100 ms
VMware Mem Usage	OK	2013-04-17 11:46:26	1d 0h 6m 8s	1/3	CHECK_VMWARE_API.PL OK - mem usage=39.30 %
VMware NET Usage	OK	2013-04-17 11:49:12	1d 0h 3m 22s	1/3	CHECK_VMWARE_API.PL OK - net usage=4061.00 KBps
VMware Runtime Issues	OK	2013-04-17 11:51:06	5d 20h 53m 31s	1/3	CHECK_VMWARE_API.PL OK - No config issues
VMware Runtime Status	OK	2013-04-17 11:49:56	5d 20h 52m 38s	1/3	CHECK_VMWARE_API.PL OK - overall status=green
VMware Swap Usage	OK	2013-04-17 11:50:50	5d 21h 1m 44s	1/3	CHECK_VMWARE_API.PL OK - swap usage=0.00 MB

Este uso en concreto tiene una pequeña pega… las passwords… En el artículo sobre macros globales comentábamos que es mejor limitar el uso de passwords al fichero resource.cfg donde se definen las macros globales. Lo estamos incumpliendo.

Lo mejor realmente sería crear en todas nuestras ILOs mediante su interface web un usuario adicional con una password única y definir estas en resource.cfg, p.e.:

```
#Acceso HP ILOs
$USER15$=UserNagios
$USER16$=password_userNagios
```

Cambiamos nuestros objetos host y command de la siguiente forma:

```
define host{
  use generic-host
  host_name virtual3
  alias virtual3
  address 192.168.0.63
  _ILO_ADDRESS 192.168.0.73
}
 # CHECK_ILO, con macros a medida
 define command {
   command_name check_ilo_mod
   command_line $USER1$/check_ilo2_health.pl -u $USER15$ -p $USER16$ -H $_HOSTILO_ADDRESS$
-n
}
```

Listo…

Como vemos, las macros variables a medida pueden ser muy interesantes para pasar cualquier valor que se nos ocurra a la hora de realizar un chequeo desde un servicio de dicho host pero debemos

tener cuidado de intentar no usarlas para passwords. Si las tienes presentes seguro que te surgirá una oportunidad de usarlas que te ahorrará complicaciones y te hará la definición de objetos más clara y versátil.